UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

PLATAFORMA DE COMÉRCIO ELETRÓNICO NO PARADIGMA DE COMPUTAÇÃO EM NUVEM

projeto realizado na

INDRA Sistemas Portugal

por

Bruno Miguel Neto Pombeiro

PROJETO

VERSÃO PÚBLICA

MESTRADO EM INFORMÁTICA

2012

UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

PLATAFORMA DE COMÉRCIO ELETRÓNICO NO PARADIGMA DE COMPUTAÇÃO EM NUVEM

projeto realizado na

INDRA Sistemas Portugal

por

Bruno Miguel Neto Pombeiro

PROJETO

Projeto orientado pelo Prof. Doutor Francisco Cipriano da Cunha Martins
e por Carlos Machado Arnaldo Pinto Da Silva

MESTRADO EM INFORMÁTICA

2012

Agradecimentos

Começo por agradecer aos meus orientadores Francisco Martins e Carlos Machado, e também ao Pedro Senos que foi como um coorientador, por me terem guiado ao longo deste ano de projeto, por me terem ajudado em tudo o que necessitei e pelo vosso apoio. Obrigado.
Quero agradecer à Indra Sistemas Portugal por me ter recebido e ter dado a hipótese de efetuar este projeto. Obrigado.
Agradeço á toda a equipa do projeto pela forma como me acolheram e ajudaram no início desta fase da minha vida. Pelo à vontade em que me puseram, pela ajuda e companheirismo demonstrado e ambiente de trabalho. Agradeço em especial ao Ricardo Seabra, Marco Ventura e Marco Pereira, que foram a primeira equipa onde estive inserido dentro do projeto, a RED.
Um grande agradecimento à minha família, pais e irmã, porque muito do que sou e do que consegui até hoje é graças a ela. Por me ter dado e continuar a dar nem a menos nem a mais do que necessito para crescer bem e feliz. Obrigado.
Em último mas garantidamente não menos importante, quero agradecer aos meus amigos que me acompanharam nestes cinco anos por todos os momentos que partilhámos e que vão ficar sempre marcados em mim. Agradeço em especial ao João Félix, parceiro numa grande dupla, tanto no trabalho, como na diversão ao longo destes anos, e à grande Ordem da Piñacolada, que os nossos eventos continuem por muitos e longos anos.

OBRIGADO!

Para vocês.

Resumo

O negócio da contratação eletrónica baseia-se na oferta de serviços através de navegadores. Por diversos motivos, pode não ser viável ao utilizador estar a preparar uma proposta através do navegador, que implica que no momento necessite de estar conetado à Internet. Também pode acontecer algum infortúnio e, num momento importante, a plataforma eletrónica estar indisponível, impossibilitando o utilizador de submeter a proposta dentro do prazo em que o procedimento está aberto a propostas.

O objetivo deste projeto foi enfrentar estes e outros desafios através do desenvolvimento de uma solução para interagir com a plataforma eletrónica mesmo quando esta não esteja disponível. Foi implementado uma aplicação *Desktop* que não necessita de instalação e que funciona em Windows, Linux e Mac. A aplicação disponibiliza ao utilizador a possibilidade de submeter ficheiros que ficam disponíveis aos utilizadores da mesma empresa. Estes ficheiros podem ainda ser assinados antes da submissão. A aplicação permite também a submissão de propostas a procedimentos recebidos pela empresa a que o utilizador pertence. O utilizador anexa documentos com a informação sobre a proposta e submete tanto os documentos como a proposta em si. Neste caso, o utilizador pode assinar e cifrar os documentos antes de os submeter.

Este documento descreve o trabalho realizado ao longo destes nove meses que permitiu obter uma solução inédita neste negócio.

Palavras-chave: Interoperabilidade, Serviços Web, XML, Submissão de ficheiros, Submissão *offline* de propostas

Abstract

The electronic-hiring business is based on providing services through web browsers. For many reasons, it may not be feasible for the users to prepare proposals using a web browser, since it needs, for instance, an online Internet connection. Also acts of god may happen and ,the platform be unavailable, making it impossible for the user to submit the proposal within the time that the procedure is open for proposals.

This project is purpose was to tackle these and others challenges through the developement of a solution for interacting with the electronic platform even when it is not available. We have developed a Desktop application that requires no installation and that runs on Windows, Linux and Mac. The application provides the possibility to upload files that will be available for all the users of the company. These files can also be signed before uploading. The application also allows for the upload of procedure proposals received by the company. The user attaches documents with information on the proposal and upload both the documents and the proposal itself. In this case, the user may sign and encrypt the documents before uploading them.

This document describes the work done during these nine months, obtaining a brand new solution for this business.

Keywords: Interoperability, Web Services, XML, Uploading files, Offline proposals upload

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Este relatório é parte integrante da unidade curricular Projeto em Engenharia Informática, disciplina do 2º ciclo do Mestrado em Informática, tendo a duração de nove meses. O projeto foi realizado na *Indra Sistemas de Portugal*, tendo como orientação académica o Prof. Francisco Cipriano da Cunha Martins, docente no Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa, e como coorientador da *Indra Sistemas Portugal* o Engenheiro Carlos Machado Arnaldo Pinto Da Silva.

1.1 Enquadramento Institucional

A Indra é a maior empresa de Tecnologias de Informação (TI) em Espanha e uma multinacional de referência na Europa e América Latina. Atualmente conta com filiais em cerca de 40 países, projetos em mais de 100 e com mais de 36.000 profissionais.

A *Indra* atua em dez setores principais: Indústria, Comércio e Serviços, Energia e Utilities, Telecomunicação e Média, Serviços Financeiros, Administração Pública, Saúde, Transportes Ferroviário, Vial e Marítimo, Defesa, Segurança, e Consultoria Estratégica de Negócio. A Figura 1.1 mostra estes setores principais e como servem o cliente através das áreas horizontais.

A constituição da *Indra Sistemas Portugal* foi feita em 2005 e conta atualmente com mais de 450 profissionais. Os valores Indra que todos os seus colaboradores seguem são a Determinação, o Rigor, a Sensibilidade e a Originalidade. São estes valores que permitem que a Indra chegue aos seus objetivos e continue a crescer.

Este projeto está a ser desenvolvido no cliente, que é líder de mercado em Portugal na área de plataformas de contratação eletrónica e com valores de referência no mercado internacional.

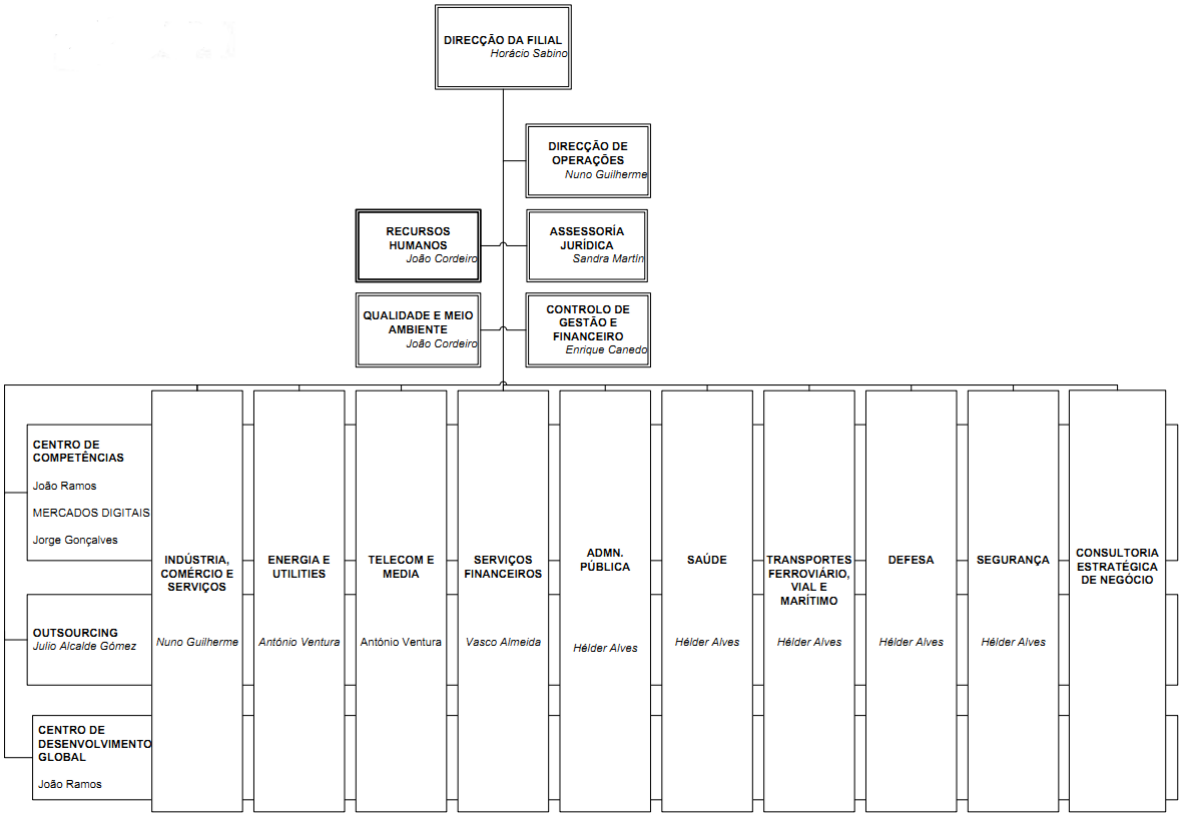

Figura 1.1: Estrutura Organizativa - *Indra Sistemas Portugal*

1.2 Integração

O meu estágio começou no dia 5 de Setembro de 2011. Na primeira semana tive uma sessão de acolhimento de novos elementos na Indra em conjunto com outros recém colaboradores. A sessão, que teve a duração de três dias, serviu para conhecer a empresa e a sua estrutura organizativa, e contou com a presença dos diretores de cada área da Indra Sistemas Portugal.

Tive também a formação *Integrate* onde foram feitas várias atividades sobre diversas áreas como a comunicação, reflexão ou trabalho em equipa. Esta sessão e formação foram muito importantes para ficar a conhecer melhor a empresa onde estou inserido e o ambiente de equipa que se valoriza. Também fiquei a conhecer melhor outros colaboradores que agora trabalham comigo no cliente, facilitando a minha integração na empresa e no projeto.

Em relação à integração no projeto, inicialmente fui apresentado à equipa onde iria ser incluido e passar os meus primeiros tempos no mundo do trabalho. Toda a equipa me ajudou pondo-me à vontade e disponibilizando um bom ambiente para uma integração fácil e rápida. Também me ajudaram bastante no conhecimento inicial sobre o projeto e as suas metodologias e tecnologias. Estas foram mesmo as minhas maiores dificuldades,

pois são utilizadas metodologias que não conhecia e algumas tecnologias que nunca tinha utilizado antes. A isto juntou-se o fato de ser um projeto de grande dimensão e de já estavar numa fase avançada. No entanto, com a ajuda de todos do projeto e com esforço e dedicação consegui adaptar-me.

1.3 Motivação

Este projeto surgiu da necessidade do cliente substituir a sua plataforma de contratação eletrónica que já tem 12 anos. A primeira plataforma foi construída apenas para o mercado privado no setor da construção, e foi desenvolvida utilizando tecnologias obsoletas e por isso com limitações. Nesta plataforma os compradores lançavam propostas e os fornecedores interessados faziam a sua proposta, que posteriormente iria ser avaliada pelo comprador. A plataforma permitia também que o processo de lançamento do concurso, de avaliações das propostas e de resposta aos fornecedores fosse efetuada eletronicamente, uma mais valia visto que na altura todas as outras plataformas de contratação eletrónica (nacional e internacional) permitiam somente que o lançamento do concurso fosse feito de forma eletrónica. Esta mais valia ajudou a empresa a crescer e a expandir o seu negócio, atuando noutros setores, além do da construção.

Na altura da expansão, a atuação no mercado público era também um objetivo, mas as limitações da arquitetura de então impossibilitavam a sua utilização neste mercado devido à necessidade de seguir certas regras impostas por lei, como por exemplo, o tipo de assinaturas de documentos eletrónicos. Para ser possível seguir estas regras e consequentemente atuar no mercado público foi criada, à parte, uma nova arquitetura que seria usada apenas neste mercado, já utilizando tecnologias mais recentes. Mais tarde, tentou-se juntar as duas arquiteturas, mas a diferença entre as tecnologias de ambas era muito grande. Este motivo, em conjunto com a necessidade de suportar novas e melhores soluções que o negócio da empresa requeria, surgiu a ideia de criar uma nova plataforma que tivesse uma arquitetura única para vários mercados e que oferecesse mais e melhores funcionalidades do que as soluções atuais. Permitiria, entre outros, um lançamento de concursos mais completo, uma avaliação das propostas recebidas mais automatizada, uma notificação dos fornecedores sobre novos concursos mais simples e uma resposta mais completa. Permitiria também uma maior facilidade na expansão da empresa internacionalmente, pois a plataforma seria adaptável às leis do país, evitando reprogramações desnecessárias para cada país onde a empresa atue. Resumindo, esta nova plataforma seria mais sofisticada e intuitiva, disponibilizando um melhor serviço aos seus utilizadores. Esta plataforma de contratação eletrónica será, daqui para a frente, denominada como plataforma eletrónica ou simplesmente plataforma.

Adicionalmente, os utilizadores terão também acesso a uma aplicação inovadora neste negócio, a *Portable Application*. Esta é uma aplicação *Desktop* que pode ser utilizada em qualquer computador sem instalação prévia e sem necessidade de instalação de nenhuma biblioteca adicional no sistema para o seu funcionamento ou permissões específicas. A *Portable Application* permite submeter documentos, ver procedimentos e submeter propostas aos mesmos independentemente da disponibilidade da plataforma eletónica no momento. Foi com este foco em mente, que surgiu a ideia de disponibilizar aos clientes esta solução que permite de uma forma simples e rápida executar tarefas mesmo que a plataforma eletrónica esteja indisponível. A *Portable Application* será, daqui para a frente, denominada simplesmente por aplicação.

Esta solução é única na área, podendo ser o primeiro passo para no futuro existirem mais aplicações e com mais funcionalidades que permitam aos utilizadores uma menor dependência do navegador e também efetuarem mais ações *offline*.

1.4 Objetivos

O objetivo deste projeto é o de desenvolver a *Portable Application*. A aplicação tem como objetivo principal permitir ao utilizador executar operações quando a plataforma eletrónica estiver indisponível. O utilizador poderá enviar ficheiros e responder a procedimentos e a aplicação suportará igualmente a autenticação, assinatura e cifra de documentos. Terá versões para os sistemas operativos Windows, Linux e Mac OS X sem necessitar de instalação prévia.

Foram definidas as seguintes necessidades da aplicação:

- submeter ficheiros à plataforma eletrónica;
- cifrar ficheiros;
- assinar ficheiros;
- resumir a submissão de ficheiros;
- ver procedimentos associados à empresa do utilizador;
- submeter propostas a procedimentos (com anexos).

De seguida serão apresentados os objetivos concretos desta aplicação, os desafios que colocam e como serão abordados.

1. **Realizar operações com a plataforma eletrónica indisponível**

   - Deverá permitir aos utilizador efetuar operações quando a plataforma eletrónica estiver indisponível;

   Desafios:

   - A aplicação tem de continuar a ter acesso a informação indispensável para a realização das operações (identificador do procedimento, a empresa que lançou o procedimento, o título, a descrição e tipo do procedimento, as datas de publicação e de limite para enviar propostas, os documentos que são obrigatórios anexar, etc).
   - As operações deverão ser realizadas de igual forma quer a aplicação se encontre num cenário *online* ou *offline*, de maneira que seja transparente para o utilizador a indisponibilidade da plataforma.

   Abordagem:

   - Na criação dos procedimentos, guardar a informação indispensável localmente de modo a que seja acessível quando necessária. Os ficheiros e propostas são submetidos para um servidor especial que processará as operações e que fica encarregado de enviar os dados para a plataforma quando esta estiver disponível.

2. **Submeter ficheiros para a plataforma eletrónica**

   - Deverá permitir aos utilizador submeterem qualquer tipo de ficheiro para a plataforma eletrónica, associado à sua empresa.
   - Deverá igualmente ser possível assinar os ficheiros antes de serem enviados e de resumir a submissão.

   Desafios:

   - Submeter ficheiros de grande dimensão podendo parar a submissão a qualquer altura, e retorná-la mais tarde a partir do ponto onde foi parada.

   Abordagem:

   - Dividir cada ficheiro em vários blocos a serem enviados em separado. Quando recebidos são adicionados aos que foram anteriormente transmitidos. Mesmo que a submissão pare, os dados são guardados e não descartados, preparando assim uma próxima submissão do mesmo ficheiro.

3. **Submeter propostas a procedimentos (com anexos)**

   - Deverá permitir aos utilizadores terem acesso aos procedimentos associados à sua empresa e criar e enviar propostas.
   - Deverá também permitir que sejam anexados ficheiros à proposta, assinar, cifrar e resumir a sua submissão.

   Desafios:

   - Distinguir os tipos de documentos que cada procedimento requer e a obrigatoriedade de ser assinado ou cifrado.
   - A data da submissão das propostas tem que ser fiável e dentro do prazo definido nos procedimentos.

   Abordagem:

   - Parte da informação indispensável guardada localmente refere-se aos tipos de documento correspondente ao procedimento e a obrigatoriedade de ser assinado ou cifrado, sendo depois lida quando se cria uma proposta.
   - O *timestamp* das propostas é criado por uma autoridade e só depois associado à proposta.

1.5 Estrutura do Relatório

Para além do capítulo introdutório, este relatório é constituído por mais cinco capítulos. O Capítulo 2 identifica e explica as tecnologias e aplicações utilizadas ao longo deste projeto. De seguida o Capítulo 3 discute a arquitetura da aplicação e apresenta os desenhos e requisitos. O Capítulo 4 apresenta as características da aplicação e pormenores sobre a sua implementação. Seguidamente, o Capítulo 5 apresenta características e módulos da plataforma eletrónica com a qual a aplicação interage. Finalmente, o Capítulo 6 apresenta a conclusão e trabalho futuro deste projeto.

Capítulo 2

Trabalho relacionado

Este capítulo descreve algumas ferramentas e tecnologias usadas na implementação da *Portable Application*. Permitirá também contextualizar futuras referências.

2.1 Serviços *Web*

2.1.1 Descrição

Os serviços *Web* são definidos como um sistema de *software* projetado para suportar a interoperabilidade em interações máquina-máquina na rede. Isto significa que esta tecnologia possibilita um elevado nível de interoperabilidade entre aplicações desenvolvidas em plataformas diferentes, graças à comunicação ser efetuada numa linguagem comum, a *Extensible Markup Language* (XML) [32].

Estes serviços utilizam a linguagem *Web Services Description Language* (WSDL) [24] para descrever a sua interface. Esta é uma linguagem baseada em XML que funciona como um contrato de serviço entre o cliente e o fornecedor, ou seja, o fornecedor oferece o serviço se o cliente efetuar um pedido de forma correta. A descrição WSDL de um serviço *Web* fornece uma descrição do serviço, de como pode ser acedido e quais os métodos disponíveis, incluindo os parâmetros que o método necessita e que estrutura de dados retorna. As estruturas de dados que são suportadas por esta linguagem podem ser de tipos primitivos ou complexos.

Os serviços *Web* identificam-se pelo seu *Uniform Resource Identifier* (URI) [22] e são baseados em tecnologias padrões como o XML e o *Hypertext Transfer Protocol* (HTTP) [3]. Estes serviços, que são disponibilizados na Web, são acedidos através do protocolo *Simple Object Access Protocol* (SOAP) [21]. Este é um protocolo para troca de informações estruturadas que usa o XML como formato das suas mensagens, para a negociação e transmissão baseia-se em protocolos como o *Remote Procedure Call* (RCP) [15] e o HTTP. O SOAP consiste num envelope que contém um cabeçalho e um corpo. Enquanto que o

cabeçalho contém as informações sobre o emissor e o receptor, o corpo contém a mensagem propriamente dita em formato XML.

O grande objetivo dos serviços *Web* é a comunicação entre aplicações (mesmo que estejam em sistemas diferentes) através da Internet. Este tipo de serviço facilita em muito o trabalho das empresas, por exemplo, se necessitarem de disponibilizar serviços específicos a parceiros. Para facilitar ainda mais a comunicação entre empresas foi desenvolvido o *Universal Description, Discovery, and Integration* (UDDI) [23], uma plataforma para o registo e procura de serviços *Web*. Um registo UDDI consiste em três componentes: informações gerais de cada organização, informações de organizações e serviços por categorias de negócios e informações técnicas sobre os serviços oferecidos pelas organizações. Esta plataforma fornece três funções: a publicação dos seus serviços por parte de uma organização, a procura dos clientes por determinados serviços, e o estabelecimento da ligação e a interação do cliente com o serviço.

2.2 *Security Token Service*

2.2.1 Terminologia

Símbolo Seguro (*Security Token*) representa uma coleção de informações de segurança como, por exemplo, decisões de autenticação ou autorização.
Contexto Seguro (*Security Context*) é um conceito abstrato que se refere ao estabelecimento de um estado de autenticação entre os intervenientes de uma comunicação.
Entidade de Confiança (*Relying Party*) refere-se a uma entidade na Internet que usa um fornecedor de identidade para autenticar um utilizador que num sistema.
Autoridade de Certificados X.509 (*X.509 Certificate Authority*) é uma entidade que emite certificados digitais de acordo com a padronização X.509 para uso de outras entidades.
***Security Token Service* (STS)** [19] é uma terceira parte numa comunicação entre um cliente e um serviço, que permite ao serviço autenticar os clientes através da emissão de símbolos seguros.

2.2.2 Descrição

A Figura 2.1 ilustra a troca de mensagens que ocorre entre o cliente, o STS e o serviço para o cliente obter um símbolo seguro para aquele determinado serviço de modo a lhe fazer um pedido.

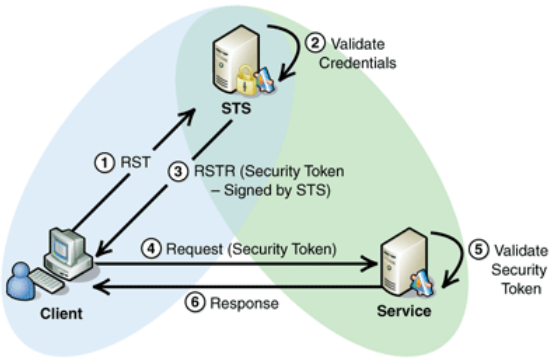

Figura 2.1: Emissão de um símbolo seguro num cenário ativo [19]

A emissão de símbolos seguros tem dois cenários: ativo e passivo. Num cenário ativo, quando o cliente precisa de efetuar, pela primeira vez, um pedido a um serviço, comunica com o STS enviando-lhe uma mensagem na forma de *Request Security Token* (RST). Estas mensagens contêm informação necessária para pedir o símbolo seguro incluindo o tipo do símbolo a ser emitido (geralmente é do tipo *Security Assertion Markup Language* (SAML), baseado em XML), as reivindicações requeridas pela entidade de confiança a serem incluidas no símbolo emitido, a informação sobre a entidade de confiança (URL e geralmente o certificado a identificar a entidade), o tempo de vida do símbolo a ser emitido e a identificação do cliente e as suas credenciais (por exemplo o nome de utilizador e a palavra-chave ou uma assinatura digital X.509).
Após receber a mensagem, o STS valida as credenciais do cliente e se estas forem válidas envia uma mensagem de resposta no formato de *Request Security Token Response* (RSTR), que contém um símbolo seguro do tipo pedido pelo cliente. Este símbolo é assinado e cifrado pelo STS para ser possível detetar alterações futuras do símbolo.
De seguida, o cliente envia para o serviço uma mensagem que contém o símbolo seguro recebido pelo STS e o seu pedido. Ao receber esta mensagem, o serviço valida o símbolo seguro para verificar que este foi emitido por um STS de confiança e que não foi alterado depois de emitido. Se for inválido ele rejeita a mensagem, caso contrário utiliza-o para estabelecer um contexto seguro com o cliente e processa o pedido do cliente. Este contexto seguro é válido enquanto for válido o símbolo seguro. Quando passar o tempo de vida o serviço já não aceita este contexto seguro. Nem sempre o cliente precisa de comunicar com o STS para pedir um símbolo seguro para um determinado serviço. O cliente só estabelece esta comunicação quando não tem o símbolo seguro para aquele determinado serviço ou quando este já ultrapassou o seu tempo de vida.

Já num cenário passivo, comum em comunicação baseada em navegadores, a mensagem RTS é partida em consultas e passada como parâmetros no URL do STS. O STS interpreta, então, esses parâmetros e processa os pedidos diretamente, ou seja, o utilizador autentica-se diretamente no STS. O símbolo é enviado na mesma ao cliente, que posteriormente envia ao serviço. Este símbolo, num cenário passivo, pode ser interpretado como uma *cookie*. Nas próximas ligações o cliente faz os pedidos ao serviço e apenas quando um pedido falhar por ter ultrapassado o tempo limite devido ao fim da validade da *cookie*, é que este redireciona a ligação para o STS de modo a que seja feito uma nova autenticação no STS e se obtenha uma nova *cookie*.

A Figura 2.2 ilustra a troca de mensagens, descrito acima, que ocorre entre o cliente, o STS e o serviço quando o cliente necessita de um símbolo seguro para aquele determinado serviço. Neste caso, antes da troca normal de mensagens para a emissão do símbolo seguro existe outra troca de mensagens de modo a estabelecer uma comunicação segura.

Nota: Nesta figura só é mostrado o estabelecimento de uma comunicação segura entre o cliente e o STS. Para haver também o estabelecimento da comunicação segura entre o cliente e o serviço basta proceder de forma idêntica ao que é ilustrado na figura, antes do passo 7, e como já foi descrito acima.

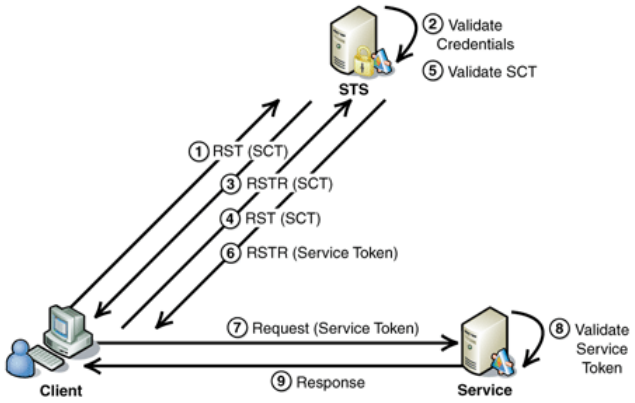

Figura 2.2: Emissão de um símbolo seguro num cenário ativo usando comunicação segura [19]

As mensagens trocadas entre o cliente e o STS e entre o cliente e o serviço contêm informação privada como palavras-chave, chaves para cifra/assinatura, entre outros, que deve ser protegida e, portanto, toda esta comunicação deve ser segura de modo a prevenir ataques maliciosos seja de leitura da informação roubada ou outros tipos de ataque. É possível alterar o modo de funcionamento padrão para que exista esta segurança na comunicação. Um exemplo de uma extensão é a alteração da troca de mensagens de modo a estabelecer uma comunicação segura entre os intervenientes.

Nesta extensão o cliente obtém do STS um *Security Context Token* (SCT) que prova a autenticação do cliente. O cliente pode usar este SCT para fazer o pedido do símbolo seguro para o serviço desejado em vez de estar sempre a enviar as credenciais originais. O STS, neste caso, só emite o símbolo seguro se a validação do SCT recebido for positiva. A sessão estabelecida entre o cliente e o serviço torna-se mais eficiente graças à utilização do *WS-SecureConversation* [29]. Esta especificação permite a criação do contexto seguro entre os dois intervenientes da comunicação. Utiliza chaves de sessão para cifrar e assinar as mensagens trocadas fornecendo assim uma forte segurança.
O cliente começa então por enviar ao STS um pedido de autenticação cujo conteúdo é o mesmo que é utilizado no padrão base. O STS, após receber o pedido, confirma a identidade do cliente com um agente de autenticação, como por exemplo, uma autoridade de certificados X.509, ou outro STS. Se se confirmar a autenticidade do cliente, então o STS emite e envia um SCT para o cliente. Este pode utilizá-lo para fazer pedidos ao STS sem enviar as credenciais originais cada vez que quer fazer um pedido. Para iniciar uma comunicação com o serviço, o cliente envia o SCT para o STS especificando o serviço a que se quer conectar. O STS valida o cliente e se este tiver autorização para aceder ao serviço, envia-lhe o símbolo seguro. Após receber o símbolo seguro o cliente estabelece uma comunicação segura com o serviço da mesma maneira que estabeleceu com o STS, ou seja, envia um pedido por um SCT usando o símbolo seguro recebido do STS de modo a que o serviço possa autenticá-lo e, como no padrão base, verifica se o símbolo seguro foi emitido por um STS de confiança e se não foi alterado depois da emissão. O serviço envia então o SCT ao cliente que o usa para fazer o pedido enviando em conjunto o SCT e o símbolo seguro. Após a validação, o serviço cria o contexto seguro com o cliente e processa o pedido do cliente, respondendo-lhe posteriormente.

2.3 *Eclipse Rich Client Platform*

2.3.1 Descrição

O objetivo de ter uma plataforma *rich client* foi sempre, essencialmente, possibilitar aos utilizadores processamento local a alta velocidade e permitir que estes se pudessem concentrar mais no trabalho e menos no sistema. Por oposição, os *thin clients* têm como vantagem a diminuição do custo de instalação e atualização ou de as máquinas dos clientes necessitarem apenas de um navegador, diminuindo assim a carga e o espaço necessário na máquina do cliente. No entanto, para manter esta vantagem é também necessário maior capacidade de rede para conseguir um bom desempenho.
A partir do momento em que as aplicações e os utilizadores começam a ser mais sofisticados e exigentes, começa também a haver novos requisitos que os *thin clients* não conseguem corresponder, como a possibilidade de se trabalhar desconectado da rede. Entretanto, começou a surgir cada vez mais plataformas que suportam componentes, como o

Eclipse RCP, que consegue corresponder aos requisitos que só os *rich clients* conseguiam, e ao mesmo tempo diminuir o custo de instalação e atualização que estes necessitavam, disponibilizando o melhor dos dois mundos.
A plataforma *Eclipse Rich Client* [42] tem um conjunto de caraterísticas que beneficiam os *rich clients*, a saber:

Componentes Os sistemas baseados em Eclipse são compostos por *plug-ins*. Este tipo de componentes permite que sejam instalados novos *plug-ins* em conjunto com outros *plug-ins* sem conflitos. Graças ao controlo de versões da plataforma as aplicaçoes podem evoluir ao longo do tempo apenas atualizando os seus componentes.

***Middleware* e Infraestrutura** O Eclipse RCP é essencialmente a função *middleware* que inclui estruturas como um paradigma flexível de UI (*User Interface*), aplicações extensíveis e atualizações.

Experiência de Utilizadores Nativos Inclui um *widget toolkit* de nome SWT (*Eclipse Standard Widget Toolkit*) que disponibiliza uma GUI (*graphical user interface*) para Java que permite eficiência e utiliza as estruturas UI do sistema operativo onde corre a aplicação.

Portabilidade Permite que a aplicação corra em muitos ambientes em variados dispositivos, como PC's, PDA's, smartphones, bastando apenas que contenham a JVM (*Java virtual machine*) com as bibliotecas J2ME (TM) ou de versões superiores.

Instalação e Atualização Inteligente A plataforma de componentes do Eclipse permite que os *plug-ins* sejam instalados e atualizados usando diversos mecanismos, como HTTP, sites de atualização ou simplesmente cópia de ficheiros.

Operações *offline* Devido a serem aplicações *rich client* não precisam de estar ligadas constantemente à rede, pois podem guardar a informação que necessitam localmente e assim evitar interrupções de rede.

Suporte a Ferramentas de Desenvolvimento O Eclipse disponibiliza um Java IDE que contém integrado ferramentas para desenvolvimento, testes e empacotamento de aplicações *rich client*.

Bibliotecas de Componentes A comunidade Eclipse desenvolveu diversos *plug-ins*, mas é possível que um utilizador possa desenvolver os seus próprios *plug-ins* e integrá-los numa aplicação.

2.4 *EnvRide*

2.4.1 Terminologia

XPATH é uma linguagem de programação usada para navegar através de elementos e atributos num ficheiro XML.

2.4.2 Descrição

O *EnvRide* [2] é uma aplicação baseada em XML [32] e XPATH [35], que disponibiliza um modo automático de alterar conteúdos num ficheiro XML.
Esta aplicação, que permite adicionar, alterar ou remover tanto atributos como elementos, é ideal para gerir ficheiros de configuração. Criando um ficheiro de configuração base podemos depois usar um ficheiro do formato envRide para alterar os valores do ficheiro de configuração em função das necessidades. Deste modo automatizamos o processo de alteração de valores em ficheiros XML. Além de ser uma tarefa repetitiva, pode levar a erros e omissões quando efetuada manualmente.
Um exemplo de aplicação do *EnvRide* é com ficheiros de configuração para diversos ambientes. Considerando, por exemplo, os ambientes de *Quality Assurance* (QA) e *Production* (PROD); sem esta aplicação teríamos que alterar manualmente os valores de modo a corresponderem ao ambiente desejado, QA ou PROD. Se considerarmos mais ambientes o trabalho, o tempo e a complexidade de atualização destes ficheiros aumentaria consideravelmente. O *EnvRide* automatiza este tipo de atividades, pois apenas necessitamos de um ficheiro de configuração com valores base e de um ficheiro do formato envRide onde definimos os campos e os valores que vão ser alterados no ficheiro de configuração base, para cada ambiente. Depois é so correr o comando para gerar o ficheiro de configuração para o ambiente desejado e a ferramente cria automaticamente um ficheiro novo com os valores definidos para esse ambiente. Também é possível, caso não queiramos criar um novo ficheiro, alterar outro já existente. Assim, podemos usar sempre o ficheiro de configuração base e durante o tempo de execução este vai sendo automaticamente alterado à medida que precisamos. A aplicação disponibiliza dois modos para efetuar as alterações: através da linha de comandos ou usando uma API para C# .Net 3.5 (ou mais recente) de modo a que seja utilizada na nossa aplicação.

2.5 Metro

2.5.1 Terminologia

MTOM/XOP [12] definem um método para otimizar a transmissão de dados nas mensagens SOAP. As especificações MTOM (*Message Transmission Optimization Mechanism*) e XOP (*XML-binary Optimized Packaging*) otimizam as mensagens SOAP extraindo o dados da mensagem e empacotando-os em anexos binários separados.

2.5.2 Descrição

O Metro [7] é uma biblioteca que disponibiliza uma solução para aceder e implementar serviços *Web*. Tem as seguintes caracteristicas:

Transporte Disponibiliza vários tipos de transportes e relaciona tecnologias de modo a conseguir uma conetividade eficiente entre os serviços. Os principais transportes disponibilizados são o transporte via HTTP (onde o Metro pode atuar tanto como cliente como servidor), via MTOM/XOP e via SOAP/TCP.

Fiabilidade Utilizando a especificação *WS-ReliableMessaging* [28] garante a recuperação de falhas causadas por perca de mensagens.

Transação Através das especificações *WS-Coordination* [27] e *WS-AtomicTransactions* [26], o Metro, garante transações atómicas para os serviços *Web*. Isto é, ou todas as transações têm sucesso ou todas falham, sendo que basta uma falhar para todas falharem.

Segurança O *WS-Security* [30] é uma especificação implementada pelo Metro que garante a integridade e a confidencialidade das mensagens. Com esta especificação não é necessário utilizar segurança baseada em transporte como o SSL (mas pode-se no entanto usar). O Metro utiliza também a especificação *WS-Trust* [31] como um meio para emitir, renovar e validar símbolos seguros usados pelo *WS-Security* e para estabelecer relações de segurança.

Para garantir a interoperabilidade nas tecnologias de serviços *Web*, o Metro tem um subsistema de nome WSIT (*Web Services Interoperability Technologies*) [25]. Este subsistema foi desenvolvido em conjunto pela *Sun* e pela *Microsoft* e inclui as seguintes tecnologias:

***Bootstrapping* e Configuração** Consiste em criar um cliente para um determinado serviço *Web*. É utilizado o URL do serviço *Web* para obter o seu ficheiro WSDL e através deste criar o cliente para o serviço *Web*.

Otimização de Mensagem Otimiza as mensagens SOAP dos serviços *Web* transmitidas pela Internet através da utilização de objetos binários. Deste modo, tanto o processamento das mensagens, como a largura de banda requerida para a sua transmissão são otimizados.

Mensagem Fiável Garante que as mensagens são entregues (se uma mensagem é perdida, a mensagem é reenviada até ser recebido uma garantia de que a mensagem foi entregue) e, opcionalmente, na ordem correta de envio.

Segurança Permite que seja estabelecido um símbolo seguro entre os intervenientes da comunicação. São geradas chaves de sessão que são depois usadas nas mensagens trocadas.

Capítulo 3

Arquitetura

Este capítulo descreve a arquitetura geral do sistema e a arquitetura, o desenho, os requisitos e o modelo de dados das operações principais da aplicação desenvolvida.

3.1 Arquitetura do Sistema

A *Portable Application* interage com um servidor próprio que serve de intermediário na comunicação com a plataforma eletrónica. Os ficheiros e propostas são submetidos para este servidor que guarda os dados e os coloca numa fila de espera para serem enviados para a plataforma eletrónica quando esta estiver disponível. Após o envio, os dados são guardados nas bases de dados da plataforma e ficam disponíveis para serem acedidos através da mesma.

A Figura 3.1 ilustra esta arquitetura.

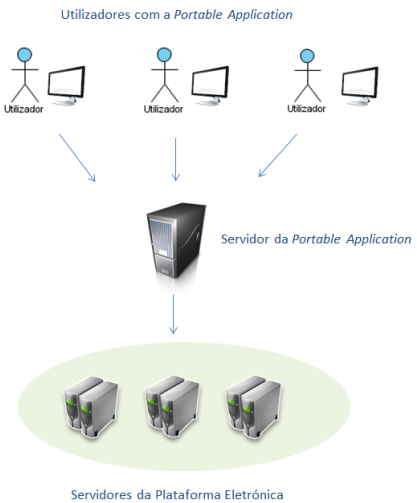

Figura 3.1: Arquitetura do Sistema

3.2 Arquitetura da Aplicação

Quando a *Portable Application* me foi atribuída, a sua arquitetura já estava maioritariamente definida. A aplicação apresentava uma arquitetura complexa e de difícil reutilização, seguindo um método de organização e responsabilidade por parte das classes pouco claro, levando a que a camada de negócio e de apresentação estivessem demasiado acopladas. Este acoplamento dificulta essencialmente a manutenção da aplicação no futuro e compromete a evolução individual e reutilização das suas camadas. Em relação à interação entre a camada de negócio e a camada de dados, responsável por ler e guardar os dados localmente, estava mais bem organizada e com uma interação mais eficiente tornando o sistema, nesse aspeto, flexível e robusto.
A Figura 3.2 ilustra as camadas lógicas.

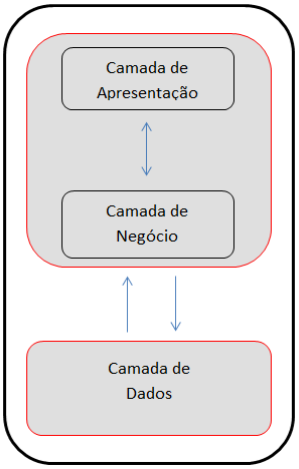

Figura 3.2: Camadas Lógicas

Nesta arquitetura, as funções específicas de determinada interface e o processamento lógico específico da mesma, encontram-se juntos. Assim, quando o utilizador faz um pedido, a própria vista que recebeu o pedido processa o mesmo e atualiza-se. Se for necessário, durante o processamento,são utilizadas as classes da camada de dados para guardar ou ler dados externos.
A figura 3.3 ilusta esta organização lógica.

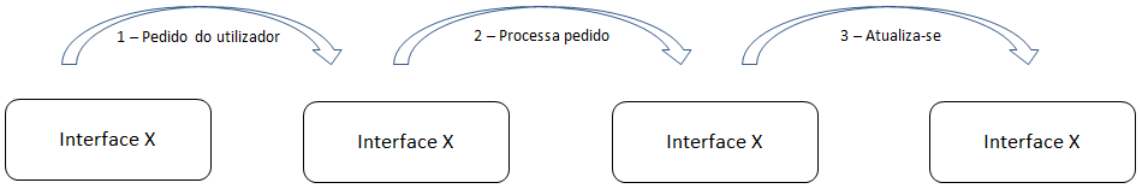

Figura 3.3: Organização Lógica atual da aplicação

Não me foi possível refazer a arquitetura devido aos prazos e planeamentos do projeto, no entanto uma melhor solução para a arquitetura desta aplicação teria sido a utilização do padrão MVC (*Model-View-Controller*) [10]. Este padrão divide a aplicação em três áreas:

Modelo Contém o processo lógico e o conteúdo específico adicional, ou seja, os objetos do conteúdo, acesso a dados externos/recursos de informação e o processamento específico da aplicação.

Vista Contém as funções específicas da interface, ou seja, representa a interface com o utilizador.

Controlador Processa e responde a eventos (ações do utilizador) e pode alterar o modelo. Define a forma como a interface reage à entrada de ações do utilizador.

A grande vantagem de utilizar esta solução é a separação entre a lógica de negócio e a lógica de apresentação, introduzindo uma camada entre os dois, o controlador. Este recebe, através da camada de apresentação, o pedido do utilizador e decide, dependendo precisamente do pedido, a operação no modelo mais indicada para o processar e reencaminha-o para essa operação. Após a finalização do processamento da operação, o controlador notifica a vista para que esta se atualize. Com esta solução, se no futuro for necessário alterar a camada de apresentação, é possível fazê-lo sem necessitarmos de alterar também a camada de lógica de negócio e vice-versa. Esta solução seria ainda otimizada através da utilização de padrões de desenho como o GRASP (*General Responsibility Assignment Software Patterns*) [41] e os descritos em GoF (*Gang of Four*) [41]. O GRASP é um conjunto de padrões que ajudam a definir a atribuição de responsabilidades a classes e o GoF é também um conjunto de padrões que estão organizados em três grupos, criação, estruturais e comportamentais. A utilização de padrões de desenho ajuda-nos a obter um sistema fácil de entender, robusto, reutilizável e extensível.

A Figura 3.4 ilustra a organização lógica descrita.

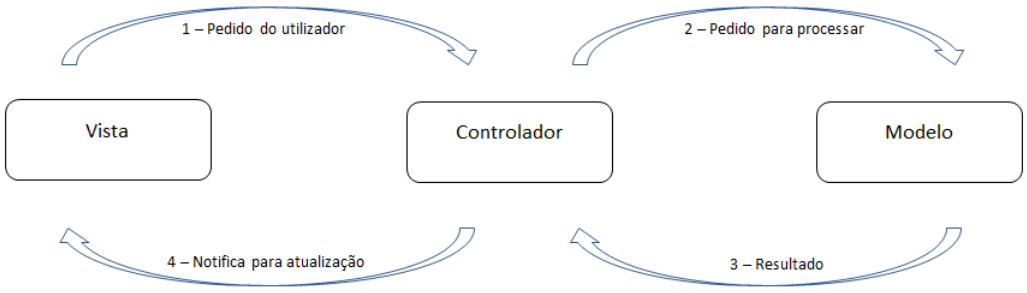

Figura 3.4: Organização Lógica de um sistema desenvolvido de acordo com o MVC

3.3 Serviços Disponibilizados

3.3.1 Assinar documentos

Âmbito e especificação do processo

Com este processo os utilizadores podem assinar os documentos a serem submetidos. Este processo é possível tanto na operação de submissão de documentos como na operação de submissão de propostas. A assinatura é feita através de um certificado selecionado previamente pelo utilizador.

Desenho e descrição

Em ambas as operações de submissão, o utilizador tem a hipótese de escolher assinar um documento de cada vez e deste modo utilizar um certificado diferente para cada documento ou assinar todos os documentos de uma só vez e com o mesmo certificado. Caso deseje assinar um de cada vez, o utilizador seleciona o documento na tabela de documentos e, por baixo desta, aparece um conjunto de informações sobre o documento selecionado. Este conjunto de informações é composto pelo nome, descrição, número de assinaturas, tamanho e estado. Aparece também um botão "Assinar" que após ser clicado procede à assinatura do documento.Se o utilizador desejar assinar todos os documentos de uma só vez, basta para isso clicar no botão "Assinar todos", disponibilizado no cimo da página.

Em ambos os casos o processo de assinatura é igual. Após clicar no respetivo botão é disponibilizada uma janela para o utilizador escolher o certificado a utilizar na assinatura. O utilizador tem à sua disponibilização dois métodos de seleção: selecionar um certificado instalado no seu computador através de uma lista de todos os certificados instalados; e selecionar o ficheiro do certificado, guardado no seu computador, através de uma janela de procura de ficheiros de acordo com o sistema operativo. Após selecionar o certificado desejado, é pedido ao utilizador que insira a palavra-chave do mesmo e de seguida a aplicação valida o certificado. Se o certificado for válido, a aplicação procede para a assinatura do documento. Caso o certificado esteja revogado ou expirado, o utilizador é informado do problema e que necessita de selecionar outro certificado para proceder com a assinatura. Após estar assinado, o estado do documento é alterado para assinado.

A aplicação disponibiliza a funcionalidade de multi-assinatura, ou seja, cada documento pode ser assinado várias vezes por certificados diferentes sem perder nenhuma das assinaturas anteriores. Esta funcionalidade é importante caso o utilizador necessite de submeter um documento assinado por um conjunto de pessoas pertencente, por exemplo, a uma divisão da empresa. O detalhe do caso de uso encontra-se de seguida.

Requisitos

Atores: Utilizador

Pré-Condições:

1. Existe pelo menos um documento para ser assinado.
2. O utilizador tem pelo menos um certificado para assinar.

Pós-Condições:

1. O documento selecionado foi assinado.

Cenário principal de sucesso:

1. O utilizador indica ao sistema que quer assinar o documento selecionado.
2. O sistema apresenta uma janela de seleção de certificado.
3. O utilizador indica se pretende selecionar através de uma lista de certificados instalados no seu computador ou selecionando o ficheiro do certificado guardado no seu computador.
4. O utilizador seleciona o certificado desejado.
5. O sistema pede a palavra-chave do certificado selecionado.
6. O utilizador insere a palavra-chave.
7. O sistema valida com sucesso que a palavra-chave inserida é correta.
8. O sistema valida com sucesso o estado do certificado selecionado.
9. O sistema assina o documento com sucesso.

Extensões:

1a. O utilizador indica ao sistema que quer assinar todos os documentos da tabela de documentos.

1. O utilizador segue os passos 2-8 do cenário principal.
2. O sistema assina cada documento com sucesso. O sistema repete o passo 2 do cenário alternativo 1a enquanto todos os documentos não estiverem assinados

1a2. O sistema não consegue assinar um ou mais documentos.

1. O sistema informa o utilizador que a assinatura do documento deu erro.

2. O caso de uso continua a partir do passo 2 do cenário alternativo 1a enquanto todos os documentos não estiverem assinados. Se já todos estiverem assinados o caso de uso continua a partir do passo 1 do cenário principal.

7a. O sistema não valida com sucesso a palavra-chave inserida.

1. O sistema informa o utilizador que introduzir uma palavra-chave incorreta.

2. O caso de uso continua a partir do passo 5 do cenário principal.

8a. O sistema não valida com sucesso o estado do certificado selecionado.

1. O sistema informa o utilizador que o certificado está revogado ou expirado e que necessita de selecionar outra certificado.

2. O caso de uso continua a partir do passo 3 do cenário principal.

9a. O sistema não consegue assinar o documento.

1. O sistema informa o utilizador que a assinatura do documento deu erro.

2. O caso de uso continua a partir do passo 1 do cenário principal.

Diagrama de Sequência de Sistema

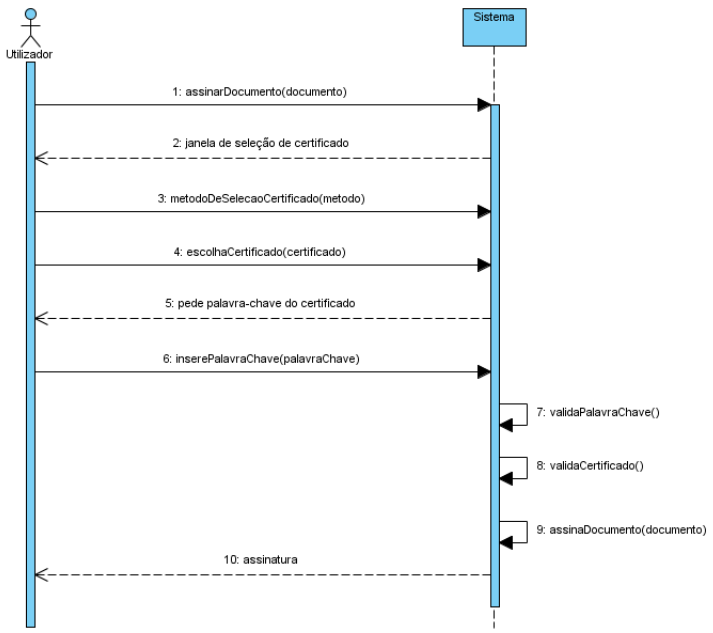

Figura 3.5: SSD do cenário principal “Assinar documento”

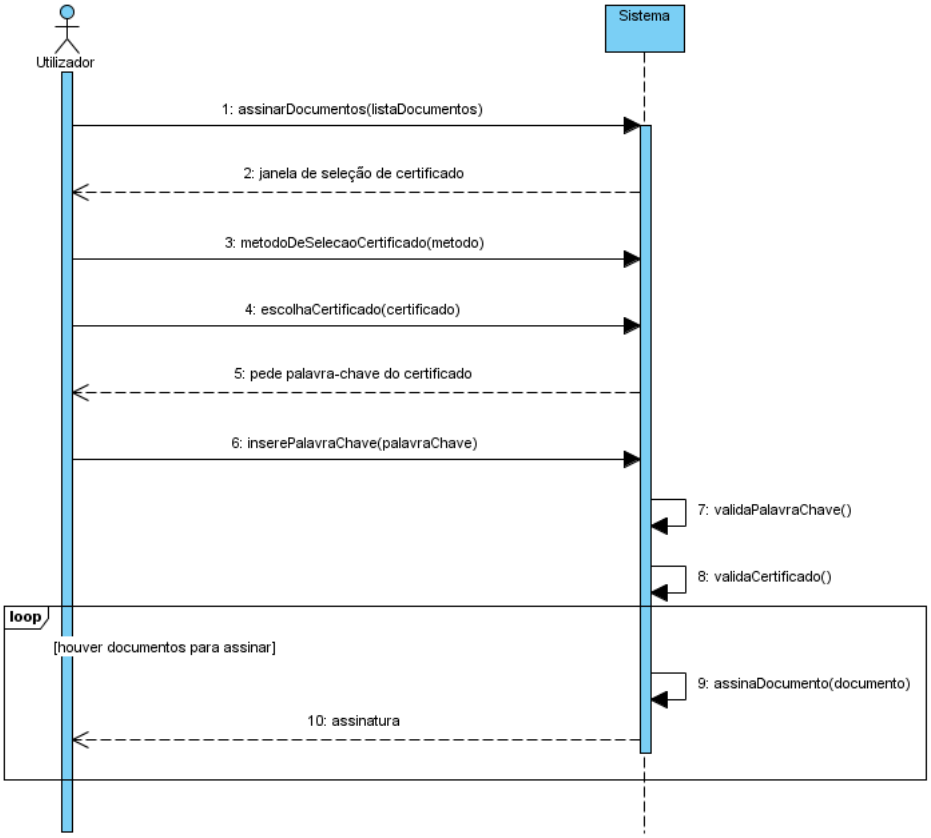

Figura 3.6: SSD do cenário alternativo 1a "Assinar todos os documentos"

3.3.2 Submeter proposta de procedimentos

Âmbito e especificação do processo

Este processo tem como objetivo permitir aos utilizadores responderem a procedimentos públicos e a procedimentos privados para os quais a sua empresa foi convidada. Os utilizadores têm a vantagem de não necessitarem de utilizar o navegador para responder e de poderem preparar a proposta *offline* antes de a submeter.

O utilizador pode escolher autenticar-se antes de criar uma proposta ou criá-la sem se autenticar. A diferença entre os dois métodos está na informação obtida e disponibilizada e na disponibilidade da plataforma para se fazer a autenticação.

Desenho e descrição

A criação de uma proposta pode ser feita de duas formas diferentes, mas com o resultado final igual. A diferença encontra-se na autenticação do utilizador.

Seguindo o método sem autenticação, o utilizador seleciona a opção de criação de uma proposta a um procedimento através das opções disponibilizadas pela interface. Ao de-

tetar que o utilizador não está autenticado, a aplicação disponibiliza uma nova janela, para o utilizador preencher, com os campos referentes ao país, identificador do utilizador, identificador da empresa e referência do procedimento a responder que correspondem a informação necessária para criar a proposta. Apenas a referência do procedimento é verificada se está correta, isto é, se existe um procedimento com a referência inserida. Se estiver incorreta o utilizador é notificado. Já os outros dados podem estar incorretos, que a proposta é criada na mesma, no entanto, após chegar à plataforma será descartada por não estar associada a um utilizador conhecido.

No caso em que o utilizador está autenticado, este pode pedir a lista de procedimentos correspondentes à sua empresa. Quando seleciona um destes da lista tem acesso à informação sobre esse procedimento e à lista de propostas desse procedimento que foram criadas anteriormente pelo utilizador (caso exista alguma). É possível nesta janela decidir criar uma proposta nova ou selecionar um rascunho que tenha sido criado anteriormente e continuar o seu preenchimento.

Independentemente do método de criação da resposta o passo final é igual. O utilizador tem à sua disposição um modelo padrão de proposta com informação sobre o procedimento e sobre os campos a preencher. Tem uma lista de tipos de documentos que variam dependendo do tipo de procedimento e aos quais pode ser obrigatório responder. Os ficheiros associados a cada tipo de documento servem como resposta e o utilizador pode anexar um ou mais ficheiros de cada tipo além de ter a hipótese de assinar ou cifrar esses ficheiros. Para certos tipos de documentos pode mesmo ser obrigatório fazê-lo, sendo que o utilizador pode ver esta informação junto de cada tipo de documento da lista.

Quando a proposta estiver pronta, basta ao utilizador submeter os ficheiros associados e no fim submeter a proposta. Tanto num caso como no outro o utilizador tem a hipótese de suspender/resumir a submissão, e se durante a submissão dos ficheiros ocorrer algum problema e parar a meio, uma nova submissão começará a transferir de onde a anterior parou. Após a submissão da proposta o utilizador tem à sua disposição o recibo da proposta. De seguida apresentamos o caso de uso.

Requisitos

Atores: Utilizador

Pré-Condições:

1. O procedimento tem que existir e estar associado à empresa do utilizador.

Pós-Condições:

1. O utilizador submeteu a proposta ao procedimento escolhido.
2. O utilizador tem acesso ao recibo da proposta.

Cenário principal de sucesso:

1. O utilizador indica ao sistema, pela lista de procedimentos, o procedimento a responder.
2. O sistema valida com sucesso a referência do procedimento.
3. O sistema apresenta o modelo da proposta para o utilizador preencher.
4. O utilizador preenche os campos da referência e da descrição da proposta.
5. O utilizador associa à proposta, um ficheiro obrigatório assinalado na tabela de anexos. O utilizador repete o passo 5 enquanto todos os tipos de documentos obrigatórios não tiverem ficheiros associados e enquanto desejar associar ficheiros e o número total de ficheiros associados for menor que 30.
6. O utilizador assina ou cifra os ficheiros associados. O utilizador repete o passo 6 até assinar ou cifrar todos os ficheiros a que são obrigados, assinalados na tabela de anexos, e enquando desejar assinar ou cifrar os ficheiros.
7. O utilizador submete os documentos e respetivas assinaturas.
8. O sistema submete os documentos para a plataforma eletrónica com sucesso.
9. O utilizador submete a proposta.
10. O sistema valida com sucesso o preenchimento dos campos da referência e descrição da proposta, que os documentos obrigatórios são associados à proposta e que os mesmos estão assinados ou cifrados caso seja obrigatório.
11. O sistema submete a proposta para a plataforma eletrónica com sucesso.
12. O sistema disponibiliza o recibo da proposta que contém as seguintes informações: o identificador do recibo, a data de criação, a empresa e o utilizador, o identificador do procedimento e da empresa que o lançou, a lista dos anexos e informação do *timestamp*, o provedor, o *timestamp* e o seu *hash*.

Extensões:

1a. O utilizador indica ao sistema, diretamente pela referência, o procedimento a responder.

2a. O sistema não valida com sucesso a referência do procedimento.

1. O sistema informa o utilizador que introduziu uma referência desconhecida.
2. O utilizador introduz uma nova a referência.
3. O caso de uso continua no passo 2 do cenário principal.

8a. O sistema não submete os documentos para a plataforma eletrónica com sucesso.

1. O sistema informa o utilizador que a submissão deu erro.
2. O caso de uso continua a partir do passo 7 do cenário principal.

10a. O sistema não valida com sucesso a proposta.

1. O sistema informa o utilizador dos erros cometidos no preenchimento da proposta.
2. O utilizador corrige a proposta.
3. O caso de uso continua a partir do passo 9 do cenário principal.

11a. O sistema não submete a proposta para a plataforma eletrónica com sucesso.

1. O sistema informa o utilizador que a submissão deu erro.
2. O caso de uso continua a partir do passo 9 do cenário principal.

Diagrama de Sequência de Sistema

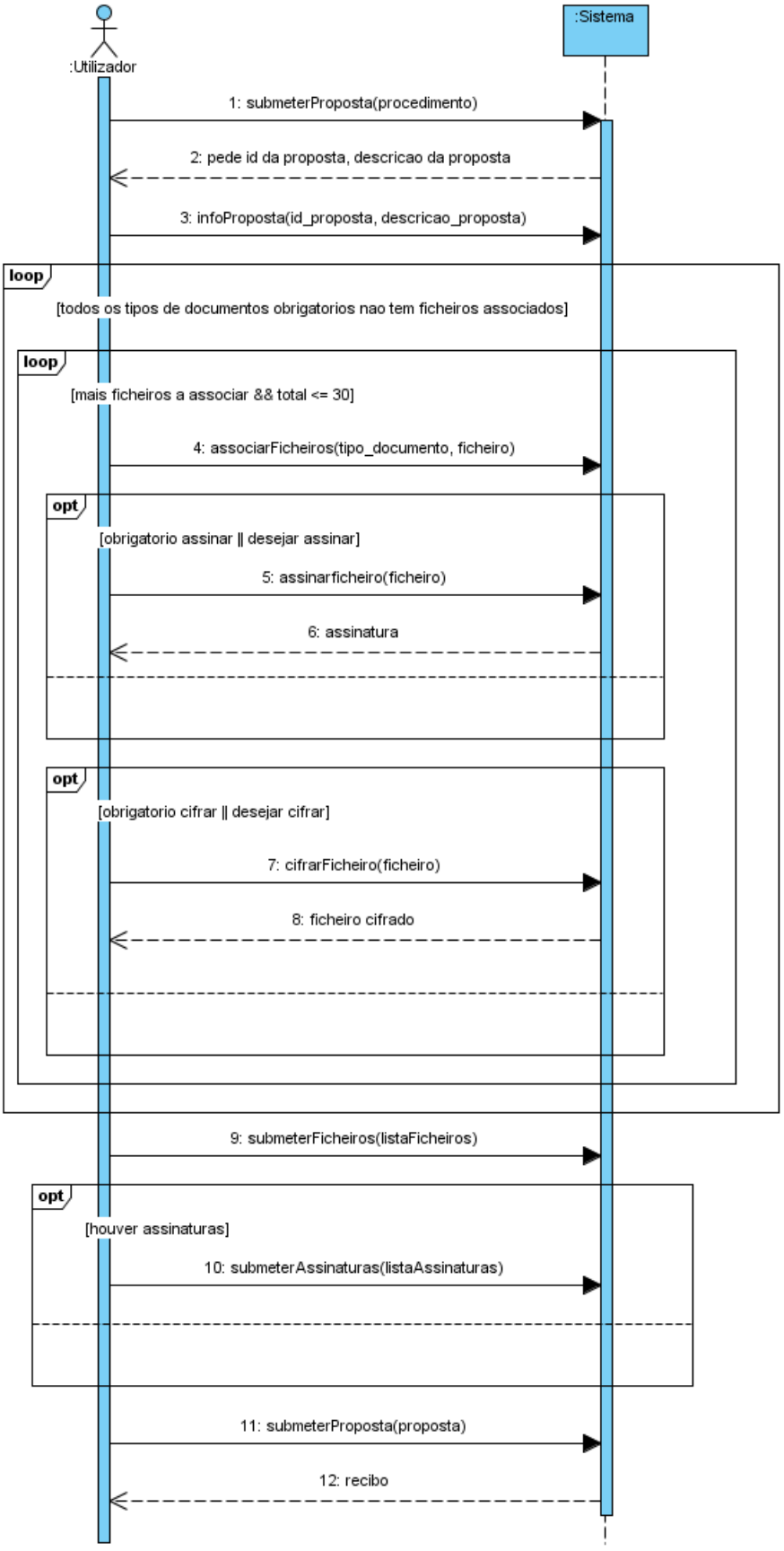

Figura 3.7: SSD do cenário "Submeter proposta de procedimentos"

3.3.3 Submeter documentos

Âmbito e especificação do processo

Este processo permite que os utilizadores submetam ficheiros para o repositório de documentos da sua empresa. Este processo pode ser particularmente útil quando um utilizador necessita urgentemente de um documento para realizar uma certa ação na plataforma e a que outro utilizador da mesma empresa tem acesso. Este último pode de uma forma fácil e rápida submeter o documento através da aplicação permitindo que o seu colega tenha acesso.

Mais uma vez o utilizador pode escolher autenticar-se ou não antes de realizar a operação. A diferença encontra-se na necessidade do utilizador preencher campos necessários para a submissão que são preenchidos automaticamente pela aplicação caso esteja autenticado.

Desenho e descrição

Como acontece com a submissão de propostas, também na submissão de documentos existem duas formas diferentes de desempenhar a tarefa. A diferença encontra-se na necessidade de o utilizador preencher certos campos de informação, que são preenchidos automaticamente caso esteja autenticado.

Após o utilizador escolher a opção de submeter ficheiros é disponibilizada uma janela com três informações, o país, o identificador da empresa e o identificador do utilizador. Caso o utilizador esteja autenticado, estas informações são preenchidas automaticamente, bastando ao utilizador adicionar ficheiros antes de submetê-los. Em situações em que o utilizador não esteja autenticado, seja por seu desejo, ou porque a plataforma não está disponível, é o próprio que tem que preencher estes três campos. A aplicação não valida os dados inseridos; como tal, se em algum campo forem inseridos valores não reconhecidos pela plataforma, esta vai descartar os ficheiros quando tentar associá-los ao utilizador e empresa através dos valores introduzidos.

É possível submeter até trinta ficheiros de uma só vez. Após serem submetidos, o utilizador terá de criar uma nova submissão se desejar submeter outros. Esta limitação acontece para evitar uma sobrecarga da aplicação e consequentemente do computador do utilizador ao submeter demasiados ficheiros de uma só vez. Existe ainda a possibilidade de assinar os ficheiros antes de serem submetidos, basta para isto o utilizador selecionar esta opção e escolher o certificado a ser utilizado. Caso o certificado seja validado com sucesso, o ficheiro é assinado e é criado um documento XML correspondente à assinatura que é submetido em conjunto com o ficheiro. Ao contrário da operação anterior, a opção de assinar os documentos é apenas do utilizador.

Durante a submissão dos ficheiros existe a hipótese de o utilizador suspender/resumir a submissão de cada ficheiro. A aplicação também está preparada para o inconveniente de por alguma razão a submissão ser parada em algum momento. Aquando de uma nova submissão dos mesmos ficheiros, esta recomeçará onde parou até submeter o ficheiro em falta totalmente. O detalhe do caso de uso encontra-se de seguida.

Requisitos

Atores: Utilizador

Pós-Condições:

1. O utilizador submeteu os documentos desejados que são transferidos para o servidor na totalidade.

Cenário principal de sucesso:

1. O utilizador indica ao sistema que quer submeter documentos.
2. O sistema apresenta o modelo da submissão para o utilizador preencher.
3. O sistema preenche os campos país, identificador da companhia e identificador do utilizador automaticamente.
4. O utilizador indica os documentos a submeter selecionando-os através de uma janela de procura de ficheiros de acordo com o sistema operativo.
5. O sistema valida com sucesso os documentos selecionados, verificando que não existem documentos repetidos. O utilizador repete os passos 4-5 enquando desejar selecionar mais documentos a serem submetidos e o número total de documentos selecionados for menor que 30.
6. O utilizador assina os documentos que desejar através da opção “Assinar Todos” ou selecionando cada documento e escolher a opção “Assinar”.
7. O utilizador submete os documentos e respetivas assinaturas.
8. O sistema submete os documentos para a plataforma eletrónica com sucesso.

Extensões:

3a. O utilizador preenche os campos país, identificador da companhia e identificador do utilizador.

5a. O sistema não valida com sucesso os documentos selecionados.

1. O sistema informa o utilizador que selecionou um documento já escolhido.

2. O caso de uso continua a partir do passo 4 do cenário principal.

8a. O sistema não consegue submeter os documentos para a plataforma eletrónica com sucesso.

1. O sistema informa o utilizador que a submissão deu erro.

2. O caso de uso continua a partir do passo 7 do cenário principal.

Diagrama de Sequência de Sistema

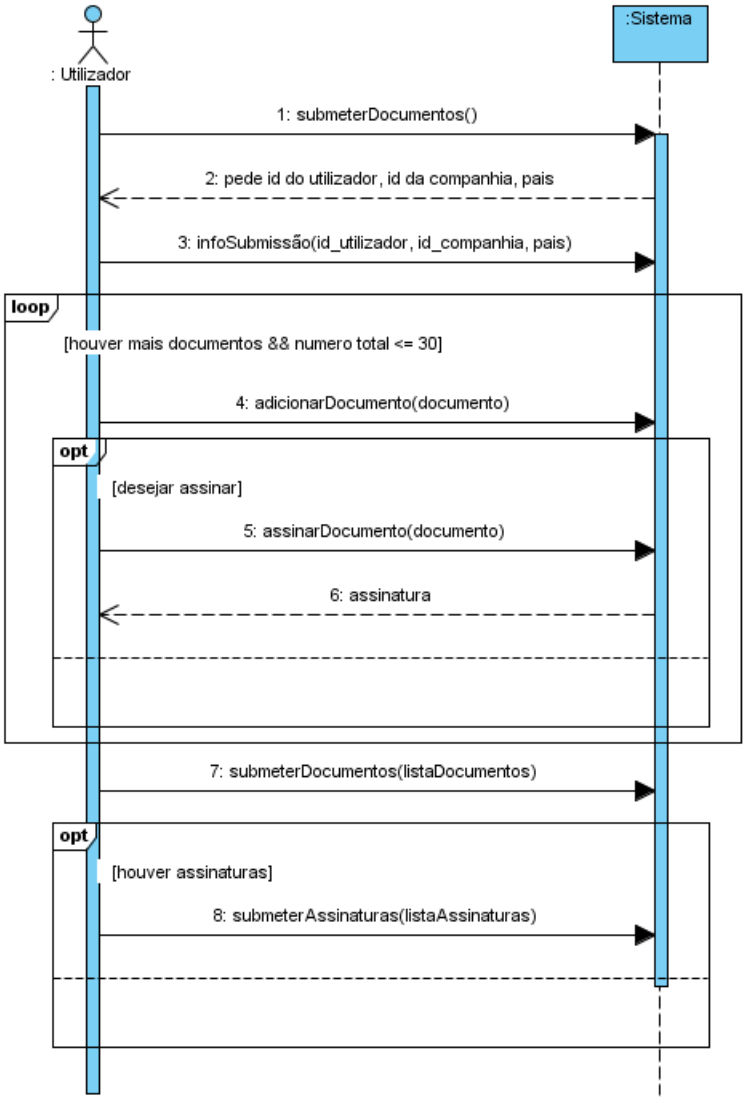

Figura 3.8: SSD do cenário "Submeter documentos"

3.4 Armazenamento de Dados

Para guardar e manter os dados há que respeitar um diagrama de armazenamento de dados para cada serviço disponibilizado.

3.4.1 Submeter proposta de procedimentos

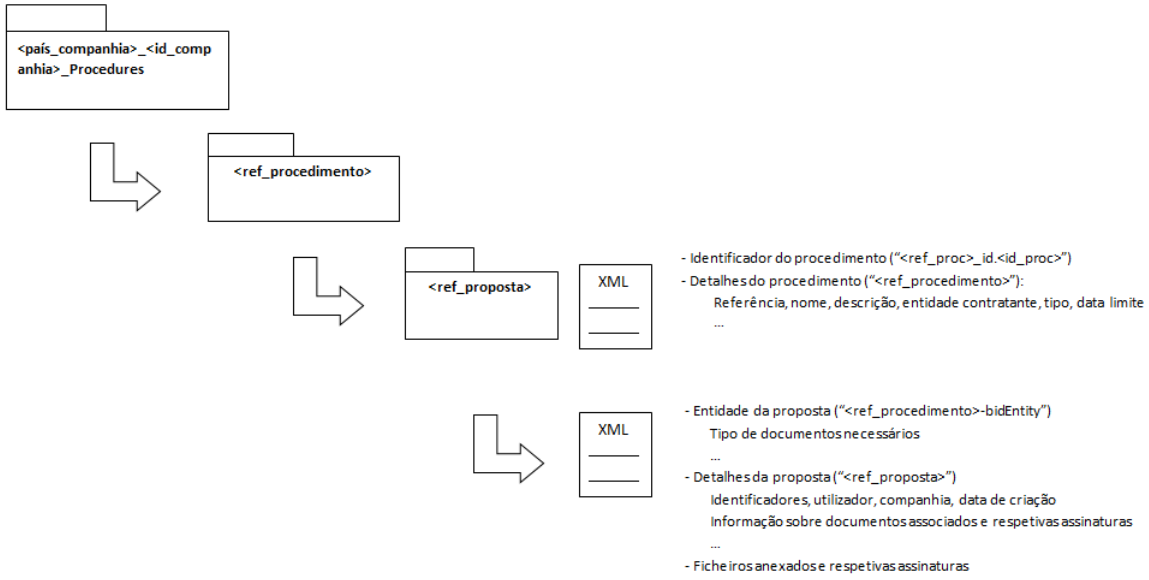

Figura 3.9: Diagrama de armazenamento de dados das propostas de procedimentos

Aquando do pedido da lista de procedimentos é criado automaticamente uma pasta onde são guardados os procedimentos da empresa do utilizador. Dentro desta pasta é criada uma pasta para cada procedimento, que conterá as informações deste e das propostas que o utilizador cria. Nesta pasta são criados dois ficheiros, um que serve de identificador do procedimento e um documento XML que contém informação sobre o procedimento. São também criadas pastas, uma para cada proposta criada pelo utilizador, onde é guardado um documento XML que é a entidade da proposta, um documento XML com a informação sobre a proposta que vai ser submetida para a plataforma e os documentos associados à proposta. Se os documentos que não estiverem no formato PDF forem assinados, são criados ainda documentos XML correspondentes à assinatura de cada documento. Caso os documentos estejam em formato PDF, as assinaturas são embebidas no próprio documento, fazendo-se antes uma cópia do ficheiro original. A informação das assinaturas é também guardada no documento XML que é submetido para a plataforma juntamente com a informação dos respetivos documentos.

3.4.2 Submeter documentos

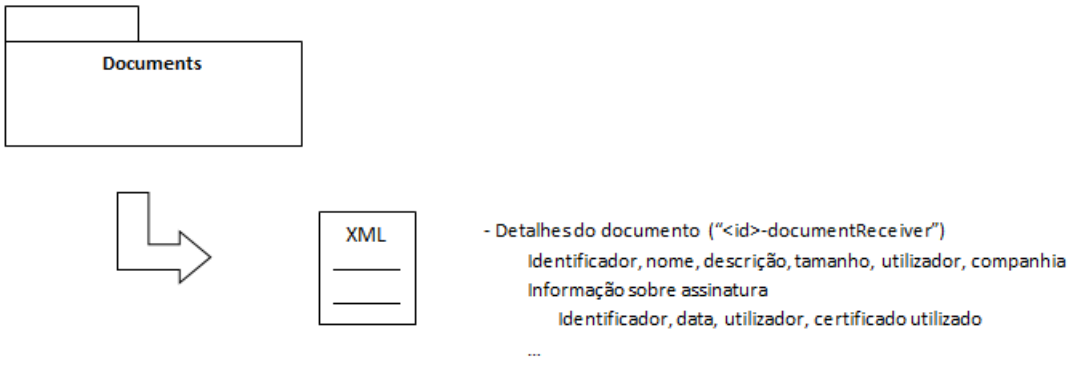

Figura 3.10: Diagrama de armazenamento de dados dos documentos submetidos

Para a submissão de documentos é criada uma pasta que vai conter os documentos XML dos ficheiros submetidos pelo utilizador. Este documento que corresponde a cada submissão, é posteriormente submetido para a plataforma juntamente com os ficheiros e contém informação sobre estes e sobre as suas assinatura, caso os documentos sejam assinados. Já os documentos XML que correspondem à assinatura, ou as cópias dos ficheiros originais caso sejam ficheiros no formato PDF, são guardados na mesma pasta onde se encontra o documento que vai ser submetido.

Capítulo 4

Portable Application

Neste capítulo descrevemos a aplicação implementada, as suas características e tecnologias utilizadas, assim como algumas alternativas que podiam ter sido seguidas.

4.1 Linguagem

A *Portable Application* é uma aplicação implementada em Java [36]. Foi escolhida esta linguagem de programação devido às suas vantagens como, por exemplo, a orientação a objetos, a independência de plataforma onde é executada e o suporte e documentação que possui.
Existem muitas outras linguagens que poderiam ter sido utilizadas para implementar esta aplicação como, por exemplo, o C [39] ou o C# [46]. No entanto, pelas razões acima referidas estas linguagens foram preteridas em relação à linguagem Java. Por exemplo, a linguagem C não é orientada a objetos e o C# é exclusivo para o sistema operativo Windows. Poderia-se usar plataformas de *software* que permitem usar o C# em Linux e Mac (um exemplo é o *Mono* [11]), mas envolveria um esforço adicional e desnecessário tendo em conta que o Java só por si já é suportado tanto em Windows como em Linux e Mac OS X.

4.2 Interface Gráfica

Para a criação da interface gráfica, a ferramenta escolhida foi o *Standard Widget Toolkit* (SWT) [45] que, além de já vir incluído no Eclipse RCP [42] (plataforma utilizada para desenvolver a aplicação), tem a vantagem de utilizar as bibliotecas do sistema operativo onde é executado. Deste modo, conseguimos que seja apresentado ao utilizador uma interface gráfica de acordo com o sistema operativo. A desvantagem de usar as bibliotecas do sistema operativo é a necessidade de a aplicação usar uma biblioteca SWT diferente para cada sistema operativo. Para superar esta desvantagem tivemos de criar uma pasta da aplicação para cada sistema operativo suportado, onde em cada uma utilizamos a biblio-

teca correspondente. Independentemente disto, o código fonte da aplicação é exatamente o mesmo. Através deste *widget toolkit* foi implementado uma interface simples e amiga do utilizador, onde é facilmente perceptível as ações que tem à sua disposição e como as realizar com sucesso.

Uma outra opção seria a utilização do SWING [40], que tem como grande vantagem em relação ao SWT ser parte integrante do java. Quando a aplicação me foi atribuída, a interface já estava praticamente toda implementada. Neste caso apenas implementei alguns componentes seguindo-me pelo que já estava feito. A interface foi baseada na aplicação de *feeds* "RSSOwl" [17].

4.3 Formato dos Dados

O *Extensible Markup Language* (XML) [32] é o formato utilizado pela aplicação para guardar os dados e para transferi-los para a plataforma. As principais razões para a escolha deste formato tem a ver com o facto de a plataforma eletrónica utilizar predominantemente XML e de facilitar a comunicação através da Internet. Para fazer a interação entre XML e Java decidimos usar a biblioteca *Java Architecture for XML Binding* (JAXB) [5] que transforma objetos Java em documentos XML e vice-versa. Como noutras bibliotecas, é criada uma árvore de objetos, com a diferença que os nós desta árvore correspondem aos elementos XML que por sua vez contêm atributos, onde o seu conteúdo são variáveis de instância. Primeiro, criamos um esquema para o documento XML e depois utilizamos as ferramentas do JAXB para gerar as classes Java e fazer as transformações. Este é escrito na linguagem de esquema W3C XML, definindo os elementos que podem aparecer no documento XML, a ordem com que aparecem, que atributos têm e o seu tipo ou que elementos contêm elementos filhos e quais. Com o esquema, e usando o *runtime* de ligação do JAXB, geramos o objeto Java correspondente ao documento XML descrito no esquema. Este objeto pode ser utilizado pela aplicação, preenchendo os seus atributos com os valores que desejamos. Quando temos o objeto preparado para ser transformado, utilizamos novamente as ferramentas da biblioteca para fazer a transformação do objeto Java para documento XML com os campos preenchidos com os valores correspondentes. É também possível transformar documentos XML em objetos Java para serem utilizados na aplicação, isto desde que o documento XML respeite o esquema definido.

Usando um exemplo, criamos o esquema "Envelope.xsd" e com o JAXB temos acesso ao objeto Java, "Envelope.java", gerado a partir do esquema. No esquema são definidos atributos e respetivos tipos como, por exemplo, a referência e descrição do envelope do tipo *String*, a data de criação em *DateTime*, o utilizador que o criou em *String* ou uma lista de documentos a serem submetidos, um tipo complexo que contém outros atribu-

tos. O objeto Java gerado contém estes e outros atributos definidos no esquema e com a aplicação preenchemos estes atributos. Por fim, utilizamos o objeto Java para criar o documento XML final, que é guardado e enviado para a plataforma eletrónica.

Duas alternativas à biblioteca JAXB são a utilização do DOM (*Document Object Model*) [1] ou do SAX (*Simple API for XML*) [20]. O DOM cria uma árvore de objectos que representa o conteúdo e organização dos dados no documento XML. Os dados ficam contidos em objetos de um simples tipo, ligados de acordo com a estrutura do documento XML. Os nós da árvore correspondem a objetos individuais que contêm atributos e cujos valores são disponibilizados sempre como *strings*. Esta árvore é criada em memória, consumindo por vezes demasiada memória e é uma solução mais lenta do que a que optámos. O SAX lê o documento e passa-o aos pedaços para a aplicação, não guardando nada em memória. É mais rápido que o DOM, mas necessita de maior esforço de programação para traduzir o XML para objetos Java.

Segue um excerto de um esquema "Envelope.xsd" utilizado na aplicação. excerto

```
1  <xs:element name="Envelope">
2    <xs:complexType>
3      <xs:sequence>
4        <xs:element name="Id" type="xs:long"/>
5        <xs:element name="CreateDate" type="xs:dateTime"/>
6        <xs:element name="EnvelopeReference" type="xs:string"/>
7        <xs:element name="EnvelopeDescription" type="xs:string"/>
8        <xs:element name="Country" type="xs:string"/>
9        <xs:element name="Username" type="xs:string"/>
10       <xs:element name="UserCode" type="xs:string"/>
11       <xs:element name="CompanyCode" type="xs:string"/>
12       <xs:element name="Type" type="xs:string"/>
13       <xs:element name="Documents">
14         <xs:complexType>
15           <xs:sequence>
16             <xs:element name="Document" maxOccurs="unbounded">
17               <xs:complexType>
18                 <xs:sequence>
19                   <xs:element name="Id" type="xs:long"/>
20                   <xs:element name="CreateDate" type="xs:dateTime"/>
21                   <xs:element name="Name" type="xs:string"/>
22                   <xs:element name="Description" type="xs:string"/>
23                   <xs:element name="Type" type="xs:string"/>
24                   <xs:element name="Size" type="xs:long"/>
25                   <xs:element name="IsSigned" type="xs:boolean"/>
26                   <xs:element name="IsEncrypted" type="xs:boolean"/>
27                   <xs:element name="Country" type="xs:string"/>
28                   <xs:element name="Hash" type="xs:string"/>
29                   <xs:element name="HashSha1" type="xs:string" nillable
                         ="true"/>
30                   <xs:element name="Storage" type="xs:string"/>
```

4.4 Comunicação com a Plataforma Eletrónica

A interação com a plataforma eletrónica é feita através de serviços *Web*. A utilização dos serviços *Web* permite a desacoplação entre a aplicação e a plataforma. Isto significa que é possível alterar a aplicação ou construir uma totalmente nova sem que isso implique alterar a plataforma. O mesmo é válido para o caso contrário, desde que as interfaces se mantenham.

Para aceder aos serviços foi criada uma biblioteca de nome "ServiceStubs". Esta biblioteca foi criada utilizando também a linguagem Java e a aplicação "Netbeans" [47], uma aplicação para desenvolvimento de *software* em diversas linguagens tal como o Eclipse. A razão da escolha desta aplicação em detrimento do Eclipse foi a maior simplicidade que apresenta na interação com serviços *Web* através do Java.
São cinco os serviços com que a aplicação interage. Quatro deles estão ligados ao STS, e portanto precisam de se autenticar perante este; e um que não necessita de autenticação. Para possibilitar a autenticação com o STS foi utilizada a biblioteca "Metro" [7]. Um exemplo de um serviço que está ligado ao STS é o serviço que é utilizado para autenticar os utilizadores. O serviço principal da aplicação, de nome "PortableService", não precisa de se autenticar e é este serviço que permite a interação com os outros serviços da plataforma e que é utilizado nas operações de submissão de documentos e de propostas. A razão de existir um serviço assim é para possibilitar ao utilizador efetuar as operações mesmo quando a plataforma não está disponível, o objetivo principal da aplicação. É possível utilizar o "PortableService" quando a plataforma está indisponível porque, ao contrário dos outros serviços que a aplicação utiliza, este encontra-se no servidor da aplicação e não nos servidores da plataforma.

4.5 Cifra

De acordo com o disposto no artigo 29º da portaria 701G de 29 de Julho de 2008, a cifra tem que ser assimétrica e com a utilização de certificados. A cifra assimétrica é uma cifra que utiliza duas chaves: uma pública e outra privada. A chave pública é acessível a todos e serve para cifrar os dados, enquanto que a chave privada pertence apenas ao destinatário e é usada para decifrar os dados anteriormente cifrados com a sua chave pública.

Quando um procedimento é publicado é também criado um certificado associado a esse procedimento por uma Autoridade de Certificados. Aquando de uma resposta, a Autoridade de Certificados é contatada para devolver a chave pública de modo a que os dados sejam cifrados. Mais tarde, na altura de o dono do procedimento aceder às respostas e decifrar os ficheiros, a plataforma contata a Autoridade de Certificados para esta lhe

disponibilize a chave privada. A autenticação entre a plataforma e a Autoridade de Certificados é efetuada automaticamente através do STS sem que o utilizador tenha alguma participação ou conhecimento. A cifra assimétrica tem duas desvantagens principais em relação à cifra simétrica: a velocidade com que é cifrado um documento e o tamanho dos dados a cifrar não pode ser maior que o tamanho da chave pública. Para contornar esta limitação, os dados são divididos em blocos do tamanho da chave e cifra-se os vários blocos em separado.
O algoritmo utilizado para a cifra assimétrica é o RSA [44], um algoritmo inventado pelos três investigadores que lhe deram o nome, que se baseia em teoria de números e na dificuldade em fatorizar números inteiros grandes.

Para acelerar o processo de cifra poderia usar-se outro método. Neste continuar-se-ia a usar a cifra assimétrica, mas para assinar a chave simétrica que seria utilizada para cifrar os dados. Isto é, gerávamos uma chave simétrica (utilizando por exemplo o algoritmo *Advanced Encryption Standard* (AES) [38]) e utilizariamos esta chave para cifrar os dados. De seguida utilizávamos a cifra assimétrica para cifrar a chave e adicionávamos o resultado ao início dos dados já cifrados. Deste modo, obtínhamos uma cifra mais rápida e igualmente segura. No entanto, teria que ser estudada de modo a garantir que não seria considerado um incumprimento da lei.

4.6 Assinatura

Na assinatura existem dois métodos diferentes, um para assinar apenas documentos PDF's e outro para assinar qualquer outro tipo de dados. A assinatura de PDF's é diferente porque é embebida no próprio PDF e como tal requer um método diferente. Para os outros tipos de dados são utilizadas assinaturas XML.

Para os documentos no formato PDF usámos a biblioteca iText [4], que permite criar e manipular PDF's de uma forma simples e rápida. O iText disponibiliza classes e métodos de modo a conseguirmos criar a assinatura do documento e de seguida inseri-la no documento PDF. A assinatura é criada através da chave privada do certificado escolhido anteriormente pelo utilizador para realizar esta operação.

Para assinar quaisquer outros tipos de dados é utilizada a especificação *XML Advanced Electronic Signatures with eXtended validation data* (XAdES-X) [33], uma especificação de assinatura XML que define formatos XML para assinaturas digitais. O XAdES-X é na verdade uma extensão da especificação *XML Advanced Electronic Signatures* (XAdES) [33], que por sua vez é uma extensão da especificação base de assinaturas XML, *XML Digital Signature* (XMLDSIG) [34]. A utilização do XAdES foi um requisito da lei.

A especificação XMLDSIG tem uma estrutura com elementos que contêm informação sobre a assinatura como, por exemplo, o método utilizado para assinar, o resumo da assinatura, as chaves utilizadas e a localização do ficheiro a ser assinado. Utilizando bibliotecas que a linguagem Java oferece, como por exemplo xml-security e xml-apis, criámos a assinatura com o formato desta especificação. O método utilizado é o RSA-SHA1 [16]. O RSA [44] além de suportar a cifra, permite também a assinatura e neste caso utiliza do algoritmo *Secure Hash Algorithm* (SHA1) [18] para criar o *hash* dos dados a assinar.

A especificação XAdES disponibiliza autenticação básica e integridade (detetar se houve alteração do documento) através de um conjunto de elementos, que descrevem propriedades da assinatura e que são adicionados à estrutura do XMLDSIG. Este nova estrutura contém elementos como, por exemplo, uma referência inequívoca do certificado que foi usado para assinar e outra da política de assinatura utilizada, e a data em que a assinatura ocorreu. Por fim, a especificação XAdES-X adiciona elementos como a lista da cadeia do certificado, a lista de certificados revogados, o *time-stamp* (gerado por uma autoridade de confiança) sobre o resumo da assinatura, e as listas já referidas. A inserção destes campos vai garantir a não-repudiação (nenhuma das duas entidades, o assinante e o verificador, pode negar a autenticidade da assinatura), e proteção caso alguma chave da cadeia do certificado ou a informação do estado de revogação esteja comprometida.

4.7 Atualização

Aproveitando as caraterísticas do Eclipse RCP desenvolvemos uma funcionalidade de atualização da aplicação fácil, rápida e sem maiores esforços para o utilizador. Deste modo, após descarregar pela primeira vez a aplicação não é necessário voltar a descarregá-la cada vez que for lançada uma nova versão. Basta apenas atualizar o *plug-in* da aplicação.

As caraterísticas do Eclipse RCP permitem que os sistemas sejam compostos por *plug-ins* e a nossa aplicação é tratada exatamente como um *plug-in*. Este é adicionado à pasta final da aplicação criada através das ferramentas disponibilizadas pelo Eclipse RCP, que contém a informação necessária para a aplicação ser executada (executável, bibliotecas)

Aquando da criação da aplicação foi definido para o executável iniciar a partir deste *plug-in* e para o atualizar basta simplesmente utilizar o *plug-in* mais recente. A aplicação tem uma opção na barra de menus para procurar atualizações e para, caso encontre, notificar o utilizador dando-lhe a opção de atualizar. Se for aceite, é descarregado o novo *plug-in* e a aplicação reinicia com a nova versão.

Utilizando as ferramentas do Eclipse RCP criámos um sítio de atualização, que tem o propósito de disponibilizar o *plug-in* mais recente da aplicação. A versão segue as normas do Eclipse e tem o formato x.x.x.YYYYMMDD-hhmm, onde os três primeiros segmentos são inteiros e são alterados dependendo da grandeza da alteração feita. O último segmento é o dia e hora da criação. Um exemplo de versão será 1.0.0.20120401-1200. Com este método é possível à aplicação determinar qual o *plug-in* mais recente e atualizar-se.

4.8 Barra de Menus

A aplicação disponibiliza também uma barra com os menus "Ficheiro", "Ferramentas" e "Ajuda". No menu "Ficheiro" o utilizador tem a hipótese de selecionar a operação de submeter documentos ou de submeter uma proposta a um procedimento, além de poder sair da aplicação. Através do menu "Ajuda" pode procurar por atualizações e ter acesso a informações sobre a aplicação. Com o menu "Ferramentas", o utilizador tem a opção "Transferência e Atividade" onde verifica o estado da transferência de atualizações, caso esteja a ocorrer alguma. Este menu disponibiliza também as preferências da aplicação. Nesta janela o utilizador pode definir as suas preferências pessoais como, por exemplo, uma *proxy*, a pasta onde a aplicação vai guardar os dados ou o idioma.

Neste momento a aplicação suporta a língua portuguesa e inglesa, mas está preparada para, de forma fácil e rápida, suportar mais línguas. A aplicação utiliza as frases escritas num ficheiro específico numa determinada língua. Quando se seleciona outra língua a aplicação simplesmente passa a ler o ficheiro correspondente. Portanto, para adicionar outra língua à aplicação basta criar um ficheiro traduzido na língua desejada e seguidamente selecioná-la nas opções.

As opções definidas pelo utilizador neste menu são guardadas num ficheiro XML chamado "UserPreferences". Este ficheiro é atualizado sempre que as opções são alteradas. Além das opções guarda outros dados como, por exemplo, a informação sobre empresas já utilizadas na aplicação, o endereço do sítio de atualização e os endereços dos serviços *Web* da plataforma a que a aplicação vai aceder.

Capítulo 5

Plataforma Eletrónica

Este capítulo apresenta caraterísticas e descreve módulos da plataforma eletrónica que são utilizados pela *Portable Application*. Descreve também a metodologia utilizada na plataforma eletrónica.

5.1 Metodologia

Foi utilizado para este projeto a metodologia *Model Driven Architecture* (MDA) [8], uma metodologia criada pela organização *Object Management Group* (OMG) [14] que se baseia na utilização e transformação entre diferentes tipos de modelos.

5.1.1 Conceitos

Antes de explicar esta metodologia é necessário introduzir alguns conceitos:

Sistema refere-se a um programa, uma pessoa, uma empresa ou um sistema de computador.

Modelo de um sistema é a descrição ou especificação desse sistema e o seu ambiente para um certo propósito. Um modelo é frequentemente apresentado como uma combinação de desenhos e texto. O texto pode estar numa linguagem de modelação ou em linguagem natural.

Ponto de vista é uma técnica para abstração utilizando um conjunto selecionado de conceitos arquiteturais e regras de estruturação, com o objetivo de nos focarmos em interesses particulares dentro desse sistema. Aqui, abstração significa o processo de suprimir determinados detalhes para estabelecer um modelo simplificado.

Plataforma é um conjunto de subsistemas e tecnologias que disponibilizam um conjunto coerente de funcionalidades através de interfaces e padrões de uso especificados. Qualquer aplicação suportada por essa plataforma pode usar essas funcionalidades

sem preocupação com os detalhes de como a funcionalidade disponibilizada pela plataforma é implementada. Exemplos de plataformas incluem sistemas operativos e a máquina virtual do Java.

5.1.2 Pontos de Vista

Ponto de vista de computação independente (CIV - *Computation Independent Viewpoint*) foca-se no ambiente do sistema e nos requisitos para o sistema, escondendo detalhes da estrutura e do processamento.

Ponto de vista de plataforma independente (PIV - *Platform Independent Viewpoint*) foca-se na operação do sistema escondendo os detalhes de uma plataforma particular..

Ponto de vista de plataforma específica (PSV - *Platform Specific Viewpoint*) combina o ponto de vista de plataforma independente, adicionando o detalhe da utilização de uma plataforma específica por parte de um sistema.

5.1.3 Modelos

Modelo de Computação Independente (CIM - *Computation Independent Model*) é uma vista do sistema na perspetiva de computação independente. É descrito como um modelo de domínio ou modelo de negócio onde se descrevem os requisitos do sistema no ambiente no qual vai operar sem, no entanto, mostrar os detalhes da estrutura e processamento do sistema.

Modelo de Plataforma Independente (PIM - *Platform Independent Model*) é uma vista do ponto de vista da plataforma independente. Exibe apenas a especificação não detalhada que se mantém de plataforma para plataforma, ou seja, exibe a especificação utilizada por diferentes plataformas do mesmo tipo sem, no entanto, exibir os detalhes dessa plataforma.

Modelo de Plataforma Específica (PSM - *Platform Specific Model*) é uma vista relativamente à plataforma específica. Combina a especificação do PIM com os detalhes que definem como o sistema utiliza um tipo particular de plataforma. É portanto um PIM com mais informação. O PSM pode ser uma implementação, se disponibilizar a informação necessária para construir um sistema e colocá-lo em operação, ou pode ser um PIM noutra transformação.

5.1.4 Transformação

O processo de transformação pode ser simples, que começa pela definição do CIM e sua transformação num PIM. De seguida, o PIM resultante da transformação é direcionado para uma plataforma e transformado num PSM. Este PSM, se disponibilizar a informação necessária para construir um sistema e colocá-lo em operação, é uma implementação. A Figura 5.1 descreve um processo simples.

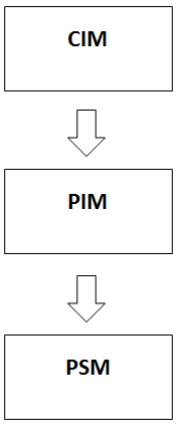

Figura 5.1: Processo simples

Em sistemas mais complexos, o processo de transformação pode levar a que ocorra mais transformações até se chegar ao resultado final. Neste caso, o PIM resultante da transformação do CIM pode ser o CIM de outra transformação, e o PSM resultante da transformação de um PIM pode ser o PIM de outra transformação. Os modelos vão sendo enriquecidos com mais detalhes até chegarem ao resultado desejado, o que no caso dos PSM é ao nível de implementação. A Figura 5.2 descreve um processo complexo.

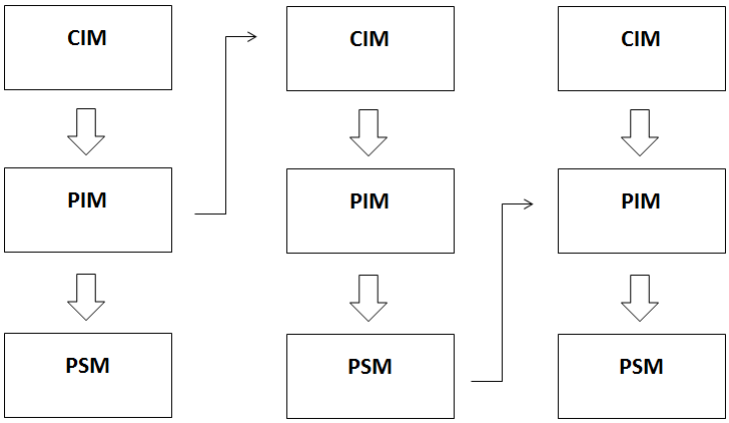

Figura 5.2: Processo complexo

Existem várias abordagens que são usadas para a transformação de modelos sendo que as principais são as seguintes:

Marcação

Neste tipo de transformação, elementos do PIM são marcados antes do modelo ser transformado em PSM e indicam o mapeamento a ser usado na transformação. Um mesmo elemento pode ser marcado diversas vezes, ou seja, pode ter marcas de diferentes mapeamentos, sendo que a sua transformação em elementos PSM depende de cada mapeamento. A Figura 5.3 descreve esta transformação.

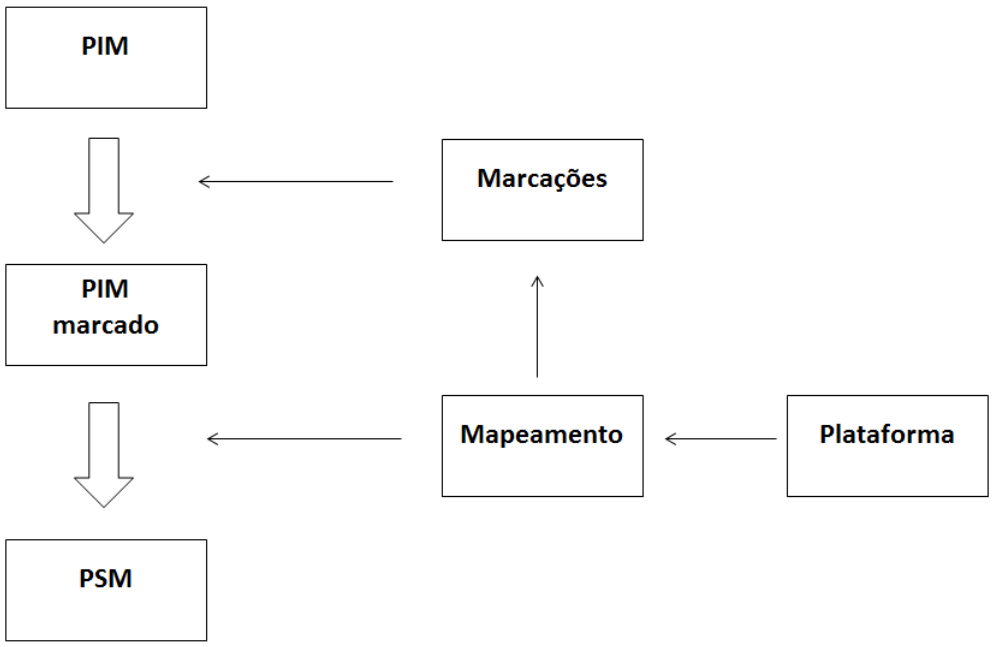

Figura 5.3: Transformação por marcação

Modelo

Neste tipo de transformação, os modelos são criados utilizando os tipos especificados num modelo. Os elementos do PIM são subtipos dos tipos especificados no modelo de tipos de plataforma independente, e os elementos do PSM, por sua vez, são subtipos dos tipos especificados no modelo de tipos de plataforma específica. As regras de transformação definidas na especificação de transformação utiliza os tipos de plataforma independente para mapear em tipos de plataforma específica. A Figura 5.4 descreve esta transformação.

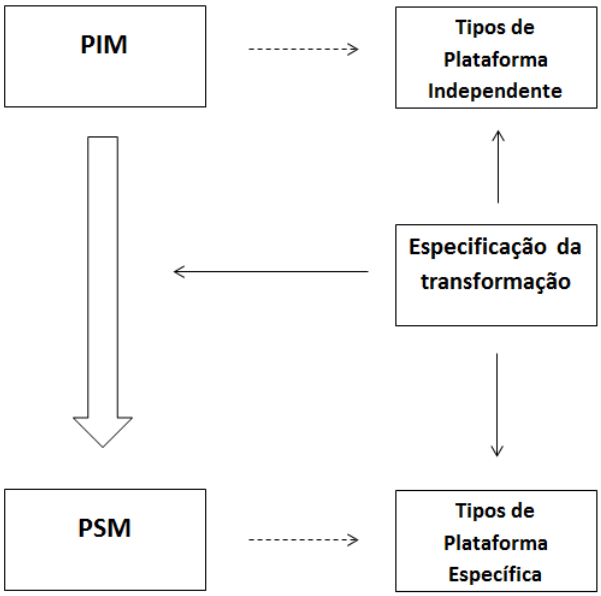

Figura 5.4: Transformação por modelo

Metamodelo

A transformação por metamodelo tem bastantes parecenças com a transformação por modelo. Neste caso, os PIMs são expressos na linguagem definida num metamodelo de plataforma independente e os PSMs são expressos na linguagem definida num metamodelo de plataforma específica. De seguida, a especificação de transformação define o mapeamento entre os metamodelos. A Figura 5.5 descreve esta transformação.

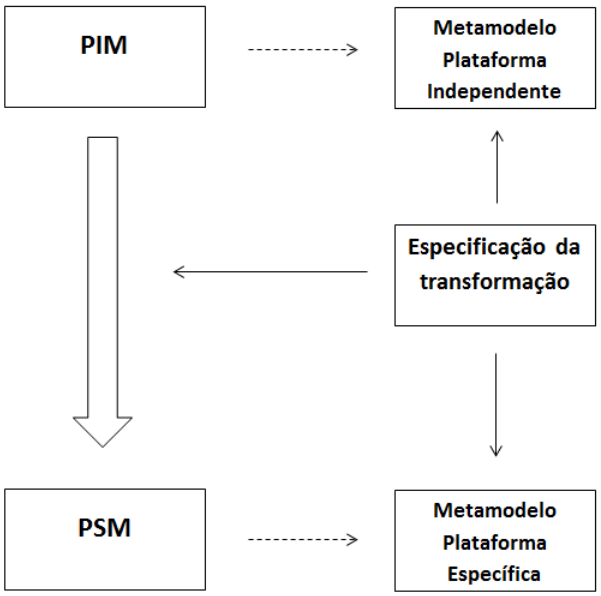

Figura 5.5: Transformação por metamodelo

Padrão

Este tipo de transformação é uma extensão de transformações por modelo e por metamodelo. Os padrões são usados para indicar grupos de elementos num modelo e mapeá-los em grupos de elementos do modelo resultante. Os tipos e padrões de plataforma independente são mapeados em tipos e padrões de plataforma específica. A grande vantagem deste tipo de transformação é que as regras de transformação podem ser aplicadas apenas a padrões específicos e não ao modelo inteiro. A Figura 5.6 descreve esta transformação.

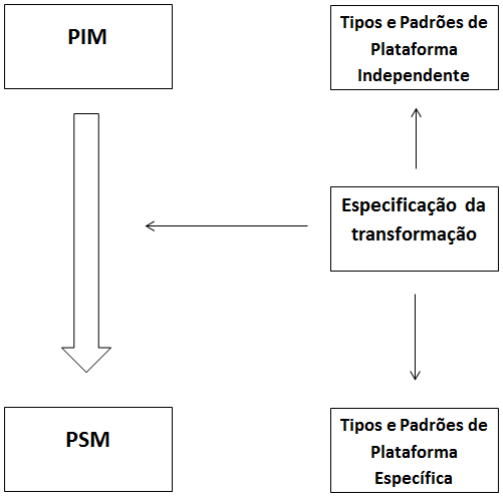

Figura 5.6: Transformação por padrão

5.1.5 Aplicação do MDA

Neste projeto, os PIMs são expressos no formato XMI (XML *Metadata Interchange*) e utilizados como argumento para a ferramenta de geração de código, feita em linguagem Java, que cria os PSMs na linguagem especificada que pode ser C# ou XML. Em certos casos os PSMs criados são posteriormente usados como PIMs noutras transformações de modo a obtermos implementações mais completas.

Esta metodologia está presente em todos as camadas da plataforma eletrónica. Desde a camada de apresentação, com a geração das classes que controlam a vista, passando pela camada de negócio, com as classes de processamento das transações de negócio e dos seus serviços, acabando na camada de dados, com a geração das classes das entidades e de acesso aos dados.
Em relação à aplicação, este metodologia foi utilizada para gerar serviços da plataforma eletrónica com que a aplicação interage, assim como para gerar classes de processamento das transações de negócio que são utilizadas no servidor da *Portable Application*.

Num projeto onde a replicação de código é necessária, o MDA permite ganhar tempo e poupar trabalho ao gerar este código replicado. Deste modo, não é necessário o programador estar a copiar o código para todas as classes onde este é necessário. Com o MDA o código é gerado nessas classes necessárias e posteriormente o programador, caso seja necessário, altera apenas certos aspetos que são exclusivos de cada classe.

Esta grande vantagem do MDA pode ser uma desvantagem em projetos pequenos e onde a necessidade de código replicado é pequena ou inexistente. O tempo perdido inicialmente a construir e atualizar a ferramenta de geração de código não compensa a pouca utilização que vai ter. Por outro lado, mesmo num projeto onde a necessidade de código replicado seja pequena esta metodologia pode ter alguma vantagem caso existam na equipa programadores que tenham experiência na utilização do MDA ou tenham já um modelo pronto a utilizar. Assim, o tempo que se iria perder a parametrizar a ferramenta de geração de código diminui consideravelmente. Em todos os casos é sempre necessário uma boa análise do projeto antes de se decidir utilizar esta metodologia.

5.2 *PortableService*

A *Portable Application* interage com um serviço principal que faz a ligação entre a aplicação e os serviços da plataforma eletrónica. É um serviço que não necessita de autenticação de modo a ser possível executar ações quando a plataforma está indisponível. Este serviço encontra-se no servidor próprio da aplicação e é utilizado nas operações de submissão de documentos e de propostas de procedimentos. O serviço é exclusivo para esta aplicação e o seu endereço não é público.

5.2.1 Métodos

De seguida são apresentados os métodos principais deste serviço. Estes métodos aqui descritos são utilizados para ter acesso a informação de procedimentos ou certificados, transferir ficheiros ou transformar objetos. Os métodos estão organizados pelo tipo de tarefas que realizam.

A Tabela 5.1 apresenta os métodos de submissão que tratam da devolução de informação dos procedimentos e da submissão de documentos e propostas.

Método	Descrição método	Parâmetro(s)	Descrição parâmetro(s)
SubmitDocReceiver	Submete a informação dos documentos submetidos	entityXML	O XML com a informação da submissão de documentos
SubmitProposal	Submete a informação da proposta ao procedimento	entityXML	O XML com a informação da proposta
GetProcedureRules	Devolve as regras do procedimento	reference	A referência do procedimento
GetProcedureRoles	Devolve a informação sobre os documentos necessários a anexar a uma proposta de um procedimento	reference	A referência do procedimento
GetCertificateData	Devolve os dados do certificado associado ao procedimento	reference	A referência do procedimento

Tabela 5.1: Métodos de submissão

A Tabela 5.2 apresenta os métodos de transferência que são utilizados para transferir os ficheiros.

Método	Descrição método	Parâmetro(s)	Descrição parâmetro(s)
PrepareToUploadFile	Prepara a submissão do documento	fileHash; fileName	*Hash* do documento; nome do ficheiro
UploadFileChunk	Submete um bloco de dados de um documento	fileNameHash; chunkIndex; chunkData; chunkDataLength	*Hash* do nome do documento; número do bloco de dados; bloco de dados; tamanho do bloco de dados
ConfirmUpload	Confirma a submissão do documento	fileNameHash; fileName	*Hash* do nome do documento; nome do ficheiro

Tabela 5.2: Métodos de transferência de ficheiros

For fim, a Tabela 5.3 apresenta os métodos de transformação XML para transformar os objetos da aplicação em objetos reconhecidos pela plataforma eletrónica e vice-versa.

Método	Descrição método	Parâmetro(s)	Descrição parâmetro(s)
PortableEnvelopeToPackage	Transforma o envelope recebido com as informações da proposta num objeto reconhecido pela plataforma eletrónica	entityXML	O XML a transformar
VortalReceiptToPortableXML	Transforma o recibo da proposta num objeto reconhecido pela aplicação	receiptXML	O XML a transformar
PortableXMLToReceiverDocument	Transforma o envelope recebido com as informações dos documentos submetidos num objeto reconhecido pela plataforma eletrónica	entityXML	O XML a transformar

Tabela 5.3: Métodos de transformações XML

5.2.2 Fluxo

Serão apresentados agora os fluxos de chamada de métodos para cada operação principal da aplicação, a submissão de documentos e a submissão de propostas a procedimentos.

Submissão de documentos

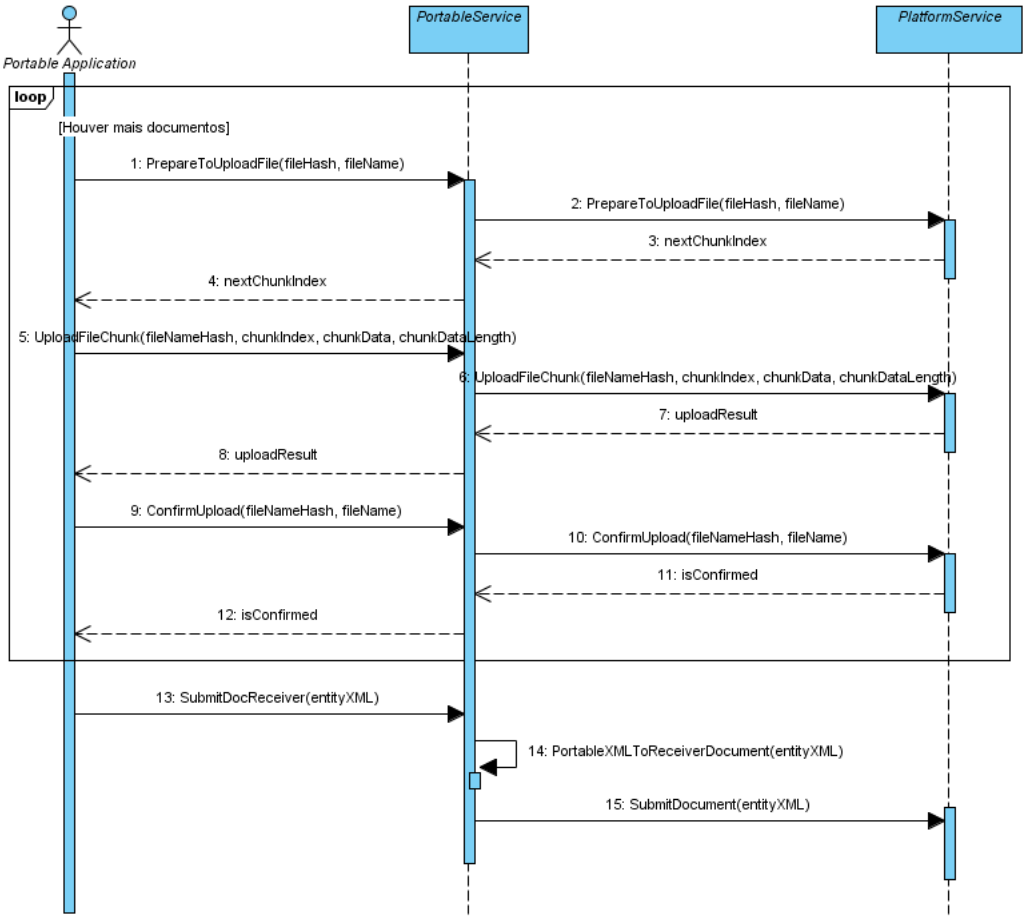

Figura 5.7: Diagrama de sequência da submissão de documentos

Como ilustra a Figura 5.7, a *Portable Application* invoca, para cada documento adicionado, um conjunto de três métodos para transferir os documentos e respetivos ficheiros de assinatura para o repositório do servidor. Posteriormente, envia o ficheiro com a informação desta submissão que inclui informação de cada documento submetido e a sua assinatura. Este método, numa primeira fase, transforma a informação recebida num objeto reconhecido pela plataforma antes de submeter o objeto resultante.

Submissão de propostas de procedimentos

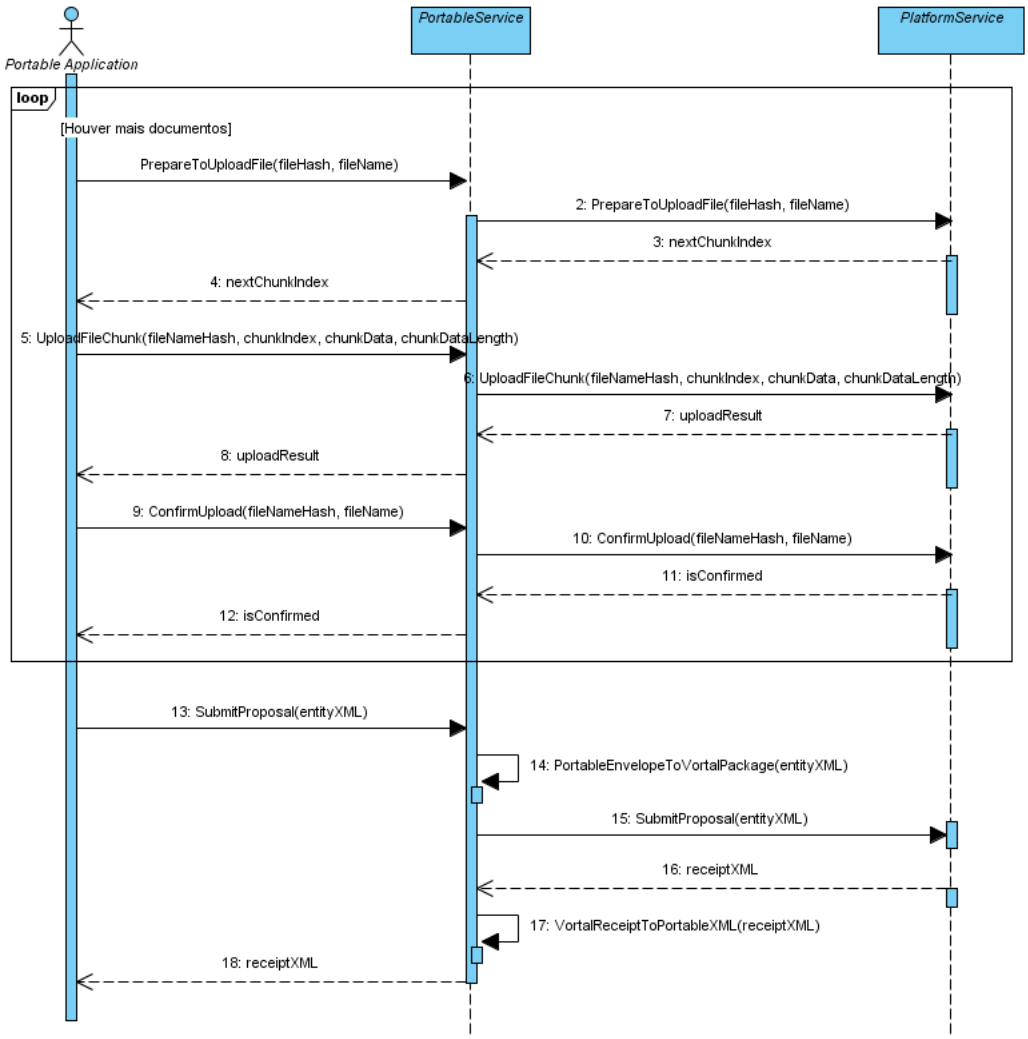

Figura 5.8: Diagrama de sequência da submissão de propostas de procedimentos

No fluxo descrito na Figura 5.8, a aplicação invoca também um conjunto de três métodos para transferir os documentos anexados à proposta. Tal como no caso anterior, os respetivos ficheiros de assinatura são transferidos para o repositório do servidor. De seguida, é submetido o ficheiro com a informação da proposta, anexos e respetivas assinaturas. No serviço é então criado o objeto, reconhecível pela plataforma, a partir da informação recebida, e posteriormente é submetido este mesmo objeto. Desta submissão é devolvido o recibo da proposta depois de ter sido transformada num ficheiro XML de modo a ser lido pela aplicação.

5.3 Comunicação

As configurações da comunicação entre serviços são definidas num ficheiro de nome "*Web.config*". Este ficheiro é um documento XML que é criado para cada aplicação *Web* com a ajuda da ferramenta EnvRide. É definido uma configuração base e para cada aplicação definimos, num documento XML, quais os valores a serem substituidos. Assim, conseguimos uma configuração própria para cada aplicação a partir de uma configuração base.

Na comunicação entre a *Portable Application* e o serviço *PortableService*, o protocolo utilizado para a comunicação é o HTTP (*Hypertext Transfer Protocol*). Por outro lado, na comunicação entre o *PortableService* e os serviços da plataforma eletrónica é utilizado o protocolo TCP (*Transmission Control Protocol*) [37] através do serviço "Net.TCP" [13], que se baseia na partilha de portas para a comunicação entre os intervenientes. A utilização deste serviço em prol do HTTPS é principalmente devido à sua maior rapidez.

A segurança da comunicação é obtida através da utilização do STS (*Security Token Service*), um serviço que autentica as entidades de uma comunicação e também estabelece uma comunicação segura entre ambas. As autenticações são feitas através do nome de utilizador e palavra-chave ou através de um certificado.

Capítulo 6

Conclusão

Ao longo deste relatório foi apresentado uma aplicação *Desktop* que possibilita a submissão de documentos e propostas a procedimentos. Esta aplicação inovadora tem como principal ponto positivo a possibilidade de continuar a realizar ações quando a plataforma eletrónica se encontra indisponível. Através de um servidor próprio, a aplicação realiza as suas ações normalmente e envia os dados para este servidor especial, onde ficam guardados até a plataforma estar novamente disponível. A ação de submeter documentos para a empresa pode ser especialmente importante no caso de elementos da mesma empresa necessitarem de trocar documentos e utilizá-los em alguma ação urgente. No caso da submissão de propostas a procedimentos, a aplicação tem uma tremenda importância nos momentos em que é necessário fazer uma proposta com o prazo do procedimento a acabar e a plataforma indisponível. Ainda possibilita que o utilizador prepare a proposta calmamente e sem necessidade de um navegador e de estar ligado à Internet, só necessitando de ligação na altura da submissão dos anexos e da proposta. Além dos pontos referidos, a aplicação desenvolvida tem ainda as vantagens de ser portátil, podendo ser guardada e carregada numa *pen*, de ser suportada em máquinas Windows, Linux e Mac OS X e de não necessitar de instalação.

Apesar destes ponto positivos, a aplicação tem também alguns pontos negativos. Um deles é ter um único ponto de falha: o seu servidor próprio. Se este servidor falhar, a aplicação deixará de conseguir buscar e submeter dados. Este ponto poderia ser resolvido utilizando mais máquinas como servidores com estado partilhado. No entanto, seria necessário ter em conta o custo de manutenção em função do número de utilizadores que utilizariam a aplicação. Aliás, este é um ponto sensível. Sendo uma aplicação inovadora num negócio que se baseia na utilização de navegadores e que envolve muito dinheiro, os clientes podem estar relutantes em utilizá-la. Terá que se introduzir a aplicação no mercado promovendo as suas vantagens através de alguns clientes selecionados de forma a que esta solução seja conhecida e aceite neste negócio.

Numa primeira fase, o trabalho futuro será tentar encontrar soluções para minorar os pontos negativos da aplicação no seu atual estado. Numa segunda fase, será importante analizar a reação dos clientes a esta aplicação e verificar junto destes a necessidade de suportar mais funcionalidades que estes considerem essenciais. Também seria interessante desenvolver um método para a aplicação funcionar sem acesso à Internet ou ao seu servidor. Em relação ao método de não necessitar de acesso à Internet, pode ter inconvenientes, nomeadamente no que diz respeito à hora de submisão das propostas. Relembro que atualmente o *timestamp* é calculado por uma autoridade e para isso é necessário acesso à Internet. Sendo possível alterar a hora do sistema operativo, é necessário arranjar uma forma de se calcular este *timestamp* de um modo fiável. Outro aspeto a analizar é se seria realmente necessário esta funcionalidade visto que neste negócio é bastante improvável que os clientes não tenham acesso à Internet na altura de submeter propostas. Já no que diz respeito a não ter acesso ao seu servidor, a solução obrigaria sempre a que fosse necessário guardar dados no computador do utilizador. Teria que ser visto a quantidade de dados que é realmente necessário guardar e as implicações que teria guardar estes dados no computador do cliente (teriam também que estar protegidas removendo as permissões de escrita). Em suma, seria sempre preciso uma análise às possíveis soluções que fossem feitas, mas o objetivo é levar esta aplicação a um nível mais elevado de modo a que cumpra as expetativas dos clientes oferecendo-lhes de um modo simples e rápido as funcionalidades essenciais que necessitam.

Abreviaturas

AES	*Advanced Encryption Standard*
CIM	*Computation Independent Model*
CIV	*Computation Independent Viewpoint*
DOM	*Document Object Model*
Eclipse RCP	*Eclipse Rich Client Platform*
GoF	*Gang of Four*
GRASP	*General Responsibility Assignment Software Patterns*
GUI	*Graphical User Interface*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hypertext Transfer Protocol Secure*
JAXB	*Java Architecture for XML Binding*
JVM	*Java Virtual Machine*
MDA	*Model Driven Architecture*
MVC	*Model-View-Controller*
OMG	*Object Management Group*
PIM	*Platform Independent Model*
PIV	*Platform Independent Viewpoint*
PSM	*Platform Specific Model*
PSV	*Platform Specific Viewpoint*
RCP	*Remote Procedure Call*
RSA	*Rivest, Shamir and Adleman*
RST	*Request Security Token*
RSTR	*Request Security Token Response*
SAML	*Security Assertion Markup Language*
SAX	*Simple API for XML*

SCT	*Security Context Token*
SHA1	*Secure Hash Algorithm*
SOAP	*Simple Object Access Protocol*
SSL	*Secure Socket Layer*
STS	*Security Token Service*
SWT	*Standard Widget Toolkit*
TCP	*Transmission Control Protocol*
TI	Tecnologias de Informação
UDDI	*Universal Description, Discovery and Integration*
UI	*User Interface*
URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locator*
WSDL	*Web Services Description Language*
WSIT	*Web Services Interoperability Technologies*
XAdES	*XML Advanced Electronic Signatures*
XAdES-X	*XML Advanced Electronic Signatures with Etended Validation Data*
XML	*Extensible Markup Language*
XMLDSIG	*XML-Signature Syntax and Processing*

Bibliografia

[1] Document Object Model (DOM). http://www.w3.org/TR/dom/.

[2] EnvRide. http://envride.codeplex.com/.

[3] Hypertext Transfer Protocol (HTTP). http://www.w3.org/Protocols/.

[4] iText. http://itextpdf.com/.

[5] Java Architecture for XML Binding (JAXB). http://jaxb.java.net/.

[6] Java Architecture for XML Binding (JAXB). http://www.oracle.com/technetwork/articles/javase/index-140168.html.

[7] Metro. http://metro.java.net/.

[8] Model Driven Architecture (MDA). http://www.omg.org/mda/.

[9] Model Driven Architecture (MDA). http://www.ibm.com/developerworks/rational/library/3100.html.

[10] Model-View-Controller (MVC). http://msdn.microsoft.com/en-us/library/ff649643.aspx.

[11] Mono. http://www.mono-project.com/Main_Page.

[12] MTOM/XOP. http://publib.boulder.ibm.com/infocenter/cicsts/v3r2/index.jsp?topic=%2Fcom.ibm.cics.ts.webservices.doc%2Fmtomxop%2Fdfhws_attachments_overview.html.

[13] Net.TCP. http://msdn.microsoft.com/en-us/library/ms734772.aspx.

[14] Object Management Group (OMG). http://www.omg.org/.

[15] Remote Procedure Call (RPC). http://www.cs.cf.ac.uk/Dave/C/node33.html.

[16] RSA-SHA1. http://www.w3.org/PICS/DSig/RSA-SHA1_1_0.html.

[17] RSSOwl. http://www.rssowl.org/.

[18] Secure Hash Algorithm (SHA-1). http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf.

[19] Security Token Service. http://msdn.microsoft.com/en-us/library/ff650503.aspx.

[20] Simple API for XML (SAX). http://www.saxproject.org/.

[21] Simple Object Access Protocol (SOAP). http://www.w3.org/TR/soap/.

[22] Uniform Resource Identifier (URI). http://www.ietf.org/rfc/rfc2396.txt.

[23] Universal Description, Discovery, and Integration (UDDI). http://uddi.xml.org/.

[24] Web Services Description Language (WSDL). http://www.w3.org/TR/wsdl.

[25] Web Services Interoperability Technologies (WSIT). http://wsit.java.net/.

[26] WS-AtomicTransaction. http://docs.oasis-open.org/ws-tx/wstx-wsat-1.1-spec-errata-os.pdf.

[27] WS-Coordination. http://docs.oasis-open.org/ws-tx/wscoor/2006/06.

[28] WS-ReliableMessaging. http://docs.oasis-open.org/ws-rx/wsrm/200608/wsrm-1.1-spec-cd-04.html.

[29] WS-SecureConversation. http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.pdf.

[30] WS-Security. https://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf.

[31] WS-Trust. http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.html.

[32] XML. http://www.w3.org/XML/.

[33] XML Advanced Electronic Signatures (XAdES). http://www.w3.org/TR/XAdES/.

[34] XML Digital Signature (XMLDSIG). http://www.w3.org/TR/xmldsig-core/.

[35] XML Path Language (XPATH). http://www.w3.org/TR/xpath/.

[36] Ken Arnold, James Gosling, and David Holmes. *The Java Programming Language*. Prentice Hall, 4th edition, 2005.

[37] Douglas E. Comer. *Internetworking with TCP/IP: Principles, protocols, and architecture*. Prentice Hall, 5th edition, 2005.

[38] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.

[39] Paul Deitel and Harvey Deitel. *C - How to Program*. Prentice Hall, 5th edition, 2006.

[40] Marc Loy Brian Cole James Elliott, Robert Eckstein and Dave Wood. *Java Swing*. O'Reilly, 2nd edition, 2012.

[41] Craig Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design*. Prentice Hall, 3rd edition, 2005.

[42] Jeff Mcaffer and Jean-Michel Lemieux. *Eclipse Rich Client Platform*. Addison-Wesley, Estados Unidos, 2006.

[43] Joaquin Miller and Jishnu Mukerji. The mda guide v1.0.1. Object Management Group, Inc, 2003.

[44] Bruce Schneier Niels Ferguson and Tadayoshi Kohno. *Cryptography Engineering: Design Principles and Practical Applications*. Wiley Publishing, Indiana, 2010.

[45] Steve Northover and Mike Wilson. *SWT: The Standard Widget Toolkit, Volume 1*. Addison-Wesley, 1st edition, 2004.

[46] Charles Petzold. *Programming Microsoft Windows with C#*. Microsoft Press, 2002.

[47] Simeon Greene Jack Woehr Tim Boudreau, Jesse Glick and Vaughn Spurlin. *NetBeans: The Definitive Guide*. O'Reilly, 2002.